INPUT JUEGOS

COLECCIONABLE DE PROGRAMACION



Aquí aprenderás cómo diseñar tus propios juegos para ordenador; desde los más simples a los más complejos de estrategia e inteligencia. Una gran abundancia de ejemplos son la manera más agradable de introducirte en los secretos de la programación.

MOVIMIENTO Y ANIMACION

- LOS PRINCIPIOS DE LA ANIMACION
- MOVIMIENTO DE GRAFICOS
- COMO UTILIZAR LOS GRAFICOS INCORPORADOS

¿Quieres darle vida a tus juegos de programación? Empieza entonces con estos sencillos caracteres gráficos, que puedes generar a partir de la memoria gráfica de tu ordenador.

Jugar con los juegos que se venden para tu ordenador es divertido sólo hasta cierto punto. Llega un momento en que la mayor parte de la gente siente la necesidad de dar rienda suelta a su imaginación y crear programas de juegos propios.

La programación de juegos no es fácil; tienes que empezar con cosas muy sencillas e ir aumentando poco a poco en complejidad. Pero eso te ayudará a pensar con lógica y aumentará tu habilidad como programador. Y también te divertirás más.

Lo primero que tienes que aprender en la programación de juegos, aparte de los trucos del BASIC, es la técnica de la animación.

Para crear la ilusión de movimiento, el programador de ordenadores utiliza en gran parte la misma técnica que el caricaturista que anima una película de dibujos. Lo que hace es crear dos (o más) imágenes y alternarlas rápidamente (idealmente, unas 24 veces por segundo).

Pero existe una diferencia importante. En la animación de dibujos, el dibujante cuenta con el proyector de películas, que le permite olvidarse de una imagen cuando ya no la necesita. En la animación por medio de un ordenador no ocurre esto. Como no hagas algo para evitarlo, cualquier segmento de imagen que «proyectes» sobre un área dada de la pantalla permanecerá allí indefinidamente.

Una forma de olvidarse de la imagen que ya no se necesita es, simplemente, imprimir algo sobre ella. El ordenador no puede situar dos imágenes al mismo tiempo en la misma posición de la pantalla.

Así si, por ejemplo, la línea 10 de un programa cualquiera dice al ordenador que imprima una A en una determinada posición, cualquier otra línea de programa que imprima (por ejemplo) una B en la misma posición se desembarazará de la A.

Los programas que siguen contienen varios ejemplos de esta clase de sustitución.

¿Pero qué ocurre si no tienes nada



que imprimir encima del carácter no deseado? Acuérdate en ese caso de incluir en alguna línea posterior una instrucción que imprima un espacio en blanco en la correspondiente posición de la pantalla.

Si te olvidas de este detalle, tu pantalla pronto se verá abarrotada con trozos no deseados de brazos, piernas y cuerpos.

La manera de obtener los caracteres gráficos en la pantalla difiere mucho de un ordenador a otro. Hay diferencias entre los caracteres gráficos incorporados en ROM y lo mismo ocurre con la forma en que se imprimen (con PRINT) en pantalla. Por último también hay diferencias en el modo de hacer que se muevan.

Esta versión de un ciempiés para **MSX** utiliza exactamente los mismos signos de máquina de escribir que otras máquinas, pero el método de generarlos sobre la pantalla es diferente. Intenta introducir el siguiente programa:

```
)))      )))      )))      )))
ooo<     ooo<     ooo<     ooo<
)))      )))      )))      )))
```

```
10 CLS
20 PRINT")))"
30 PRINT"ooo<"
40 PRINT")))"
50 LOCATE0,0
60 PRINT"((("
70 PRINT"ooo<"
80 PRINT"((("
90 GOTO10
```

Cuando lo ejecutes (RUN) observarás una imagen rápidamente cambiante. Se debe a los conjuntos separados de símbolos de las sentencias PRINT que se superponen unos a otros. Al mismo tiempo, la sentencia GOTO de la última línea crea un bucle continuo: le dice al ordenador que vuelva a empezar.

Sin las líneas 10 y 50 el programa no podría funcionar adecuadamente. En ambas líneas se hace uso de instrucciones de movimiento de cursor y de borrado de pantalla, que combinadas con la instrucción PRINT permiten llevar a cabo la animación.

¿Cómo funcionan? La instrucción LOCATE 0,0 de la línea 50 coloca el cursor en la esquina superior izquierda de la pantalla (línea 0, columna 0). Esto quiere decir que toda actividad de impresión posterior comienza en esta posición, de modo que todos los caracteres que se imprimen, desde la línea 60 en adelante, lo hacen encima de los que ya estaban allí.

Por su parte la instrucción CLS de la línea 10 hace algo más. Después de hacer volver al cursor a la parte superior izquierda de la pantalla, borra todo su contenido, dejándola preparada para que aparezca la siguiente imagen.

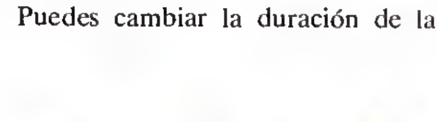
MAS DESPACIO

El movimiento del insecto es hasta ahora más bien rápido y tal vez haya que retardarlo. La manera más fácil de hacer esto es utilizar un bucle FOR...NEXT. Introduce pues la línea:

```
45 FOR T=1 TO 50:NEXT
```

Cuando pulses RETURN y ejecutes (RUN) el programa, el movimiento resultará mucho más pausado. El bucle FOR...NEXT actúa como un contador, en este caso contando hasta 100, antes de que el programa vaya a la línea 50.

Puedes cambiar la duración de la



pausa sin más que sustituir «100» por cualquier otro número de tu elección: cuanto más grande el número, mayor será el retardo.

Intenta también cambiar la posición del bucle de retardo FOR...NEXT a la línea 15. Dependiendo del retardo que escojas, habrá una apreciable pausa —una pantalla limpia— cuando se ha borrado una imagen pero todavía no ha sido sustituida por la siguiente. Por esta razón es mejor utilizar LOCATE 0,0 que CLS, dentro de un programa de este tipo.

Aunque la imagen ya está ahí, todavía resulta algo espasmódica, debido a que el bucle de retardo sólo actúa sobre la primera imagen. Se puede obtener un movimiento mucho más adecuado introduciendo en el programa otro bucle FOR...NEXT que actúe sobre la segunda imagen. Inserta pues:

```
85 FOR I=1 TO 50:NEXT
```

Este retardo es más corto, a fin de crear un movimiento «de pierna» ligeramente irregular, pero naturalmente, puedes cambiarlo si quieres.

CARACTERES EN MOVIMIENTO

El siguiente paso es alterar el programa de forma que el «cuerpo» del insecto parezca cruzar la pantalla. Para ello el BASIC MSX nos ofrece dos instrucciones que vamos a utilizar. La primera de ellas, que puede ser adecuada en aplicaciones sencillas, es la instrucción TAB. TAB siempre va seguido de un número entre paréntesis, por ejemplo TAB (15), que hace que el cursor se coloque en la columna 15 de la pantalla, o por una variable también entre paréntesis.

Además TAB siempre forma parte de la sentencia PRINT a la que se aplica. En este caso se hace uso de una variable, la variable P de la línea 10, para hacer que la posición de TAB se mueva por la pantalla. Esta variable cambia su valor al ir incluida dentro de un bucle FOR...NEXT.

```
10 FOR P=0 TO 32
20 CLS
25 PRINT TAB(P)")))"
30 PRINT TAB(P)"ooo<"
40 PRINT TAB(P)")))"
50 FOR I=1 TO 45:NEXT
55 LOCATE0,0
57 PRINT TAB(P)"((("
60 PRINT TAB(P)"ooo<"
70 PRINT TAB(P)"((("
80 FOR I=1 TO 50:NEXT
90 NEXT  P
```

Lo que has hecho es suprimir la sentencia GOTO original de la línea 90. Ahora ha sido remplazada por un bucle FOR...NEXT, que incrementa en 1 la variable P cada vez que se repite el programa. Como P forma parte de la sentencia TAB, el insecto se mueve sobre la pantalla, a razón de un paso por cada ciclo del programa.

Al ejecutar (RUN) el programa, verás que el insecto se mueve sobre la pantalla y se para al llegar al lado de-



recho. Para que la acción comience de nuevo necesitas agregar:

```
100 GOTO 10
```

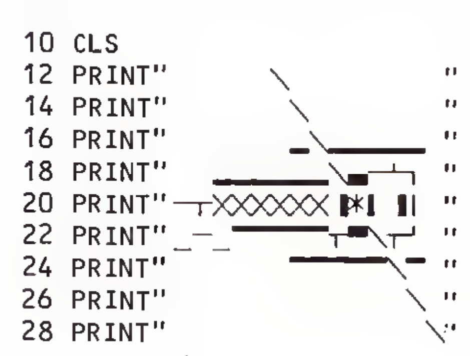
La otra instrucción es LOCATE X,Y que permite colocar el cursor en la columna X, fila Y de la pantalla. LOCATE se utiliza normalmente antes de la sentencia PRINT. En ese caso se ha utilizado LOCATE con el valor P para la columna, y los valores 7,8 y 9 para la fila. Estos tres valores corresponden a cada una de las partes de nuestro insecto, que se imprimen en filas consecutivas. La utilización de LOCATE en este ejemplo tan sencillo resulta casi más complicada que la de TAB. Pero hay que pensar que TAB sólo permite desplazar el cursor dentro de una línea y no de una línea a otra. Por ello, en caso de movimientos más complejos siempre será preferible usar LOCATE.

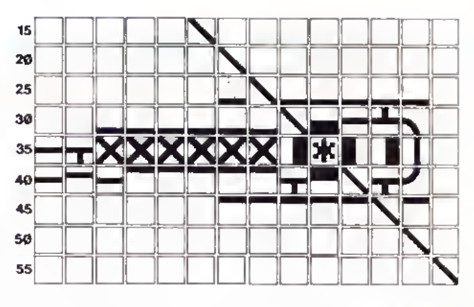
```
10 FOR P=0 TO 32
20 CLS
25 LOCATEP,7:PRINT")))"
30 LOCATEP,8:PRINT "ooo<"
40 LOCATEP,9:PRINT ")))"
50 FOR I=1 TO 45:NEXT
57 LOCATEP,7:PRINT "((("
60 LOCATEP,8:PRINT "ooo<"
70 LOCATEP,9:PRINT "((("
80 FOR I=1 TO 50:NEXT
90 NEXT  P
```

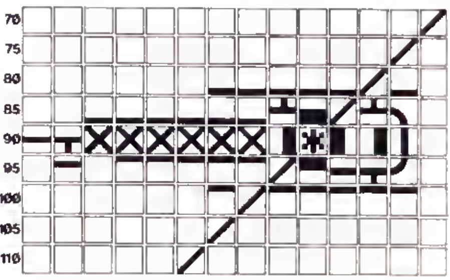
GRAFICOS EN ROM

El insecto nos ha ayudado a comprender algunos de los fundamentos de la animación, pero como gráfico resulta bastante simple. Para conseguir algo más elaborado podemos utilizar los caracteres gráficos de la memoria ROM. Todos los **MSX** disponen de estos caracteres gráficos a los que se puede acceder desde el teclado. Para ello no hay más que pulsar la tecla [GRPH], o bien las teclas [SHIFT] y [GRPH] simultáneamente. En la pantalla habrán aparecido numerosos símbolos gráficos que podrás combinar a tu gusto. Para que te hagas una idea de lo que puedes conseguir, observa el gráfico del helicóptero. Está hecho utilizando exclusivamente caracteres gráficos. Aquí tienes el programa que lo dibuja. Si quieres ver al helicóptero en acción teclea y escribe RUN.

```
10 CLS
12 PRINT"                  "
14 PRINT"                  "
16 PRINT"                  "
18 PRINT"                  "
20 PRINT"                  "
22 PRINT"                  "
24 PRINT"                  "
26 PRINT"                  "
28 PRINT"                  "
```

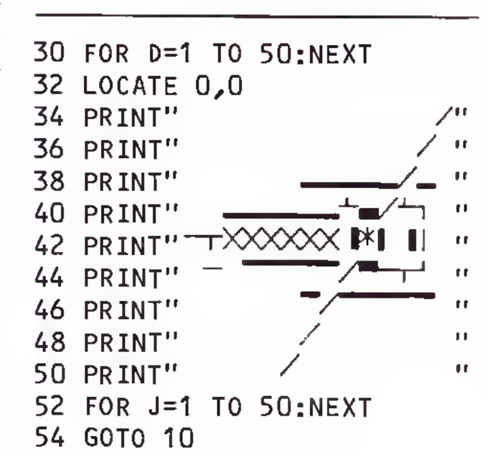






Cómo construir un helicóptero. Cada cuadro es un gráfico de ROM.

```
30 FOR D=1 TO 50:NEXT
32 LOCATE 0,0
34 PRINT"                  /"
36 PRINT"                 / "
38 PRINT"                /  "
40 PRINT"               /   "
42 PRINT"                   "
44 PRINT"             /     "
46 PRINT"            /      "
48 PRINT"           /       "
50 PRINT"          /        "
52 FOR J=1 TO 50:NEXT
54 GOTO 10
```





DERECHA...IZQUIERDA... ARRIBA...¡FUEGO!

- DETECCION DE LAS PULSACIONES DE TECLA
- LANZAMIENTO DE MISILES
- CONTROL DE UN GRAFICO MOVIL

Los juegos de extraterrestres se basan en la habilidad del jugador para controlar los sucesos de la pantalla. He aquí cómo controlar el movimiento, lanzar misiles e integrarlos en un juego.

Los juegos de marcianitos serían terriblemente aburridos si el movimiento de la base lanzadora de ráyos láser o el disparo de los mismos no se pudiera controlar de alguna manera.

El control por teclado de ese tipo de movimientos es una importante faceta de todos los juegos de marcianitos, por lo que si piensas escribir alguno, es importante comprender sus principios.

Para ello, el primer paso es hacer que el ordenador reaccione cuando pulses una tecla.

DETECCION DE LAS PULSACIONES DE TECLA

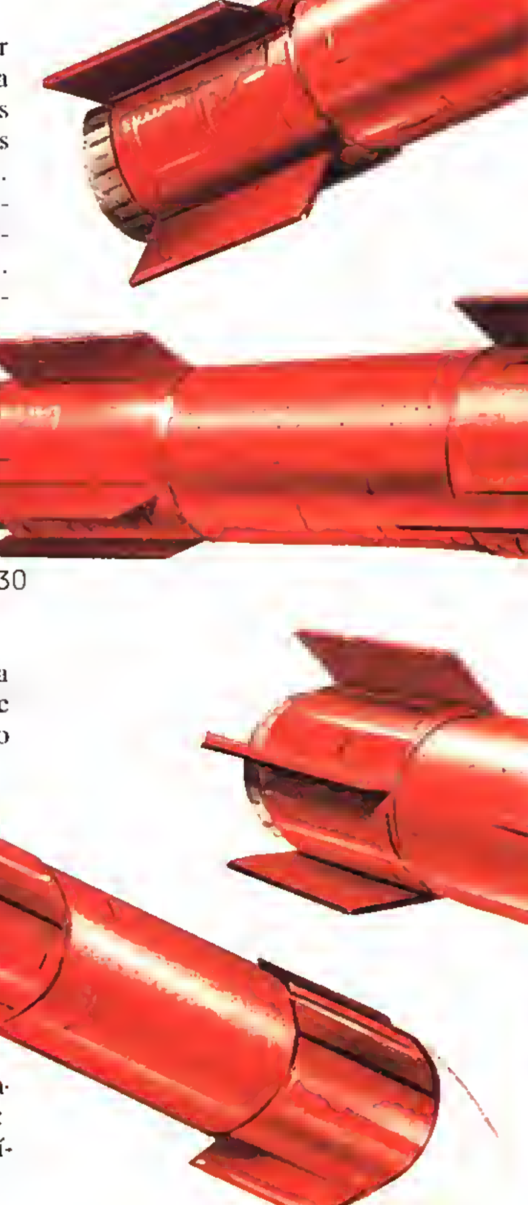
En principio todos los ordenadores domésticos se sirven del mismo método para detectar una pulsación de una tecla, si bien los detalles varian amplia mente de unos a otros.

Los ordenadores MSX pueden utilizar la sentencia INKEY$ para detectar la pulsación de cualquier tecla. Cuando el ordenador encuentra INKEY$ en un programa, explora el teclado y comprueba si hay alguna tecla pulsada. El programa que sigue es un ejemplo típico de cómo se utiliza INKEY$ para hacer esto.

```
20 CLS
30 A$=INKEY$:IF A$="" THEN 30
40 LOCATE14,10:PRINT"AUUG!"
```

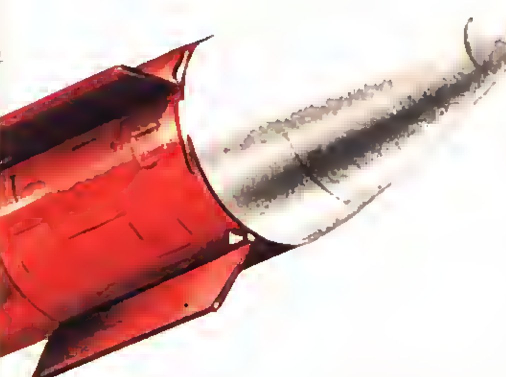
Haz correr el programa una y otra vez (con RUN) y comprobarás que pulsando cualquier tecla, excepto SHIFT, GRPH, CTRL y CODE, se origina la aparición en pantalla del mensaje AUGG!. El programa trabaja así:

La línea 20 limpia la pantalla. La línea 30 hace que el ordenador espere hasta que se pulse una tecla antes de continuar con el programa. Date cuenta que no está incluido un espacio entre las comillas. Por ello, la línea 30 significa: «Si INKEY$=nada, o si no ha sido pulsada tecla alguna, vuelve a comprobar». Es importante disponer del IF...THEN 30, por-

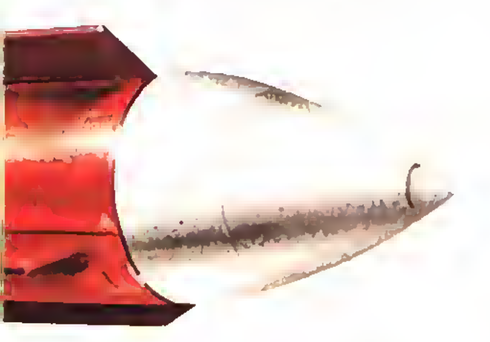


que de otra forma el ordenador comprobaría solamente una vez si ha sido presionada una tecla y sólo durante una fracción de segundo.

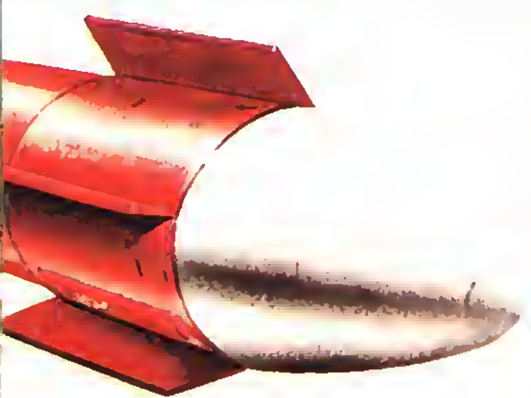
Tan pronto como la tecla sea pulsada, A$ toma el valor de esa tecla. Por ejemplo, si presionamos el 3, entonces A$=«3». Esto es suficiente para que la línea 40 visualice «AUUG!» en la pantalla.



En la mayoría de los juegos debes presionar una determinada tecla para mover un tanque, una nave espacial o cualquier otra cosa. Si alteras la línea 40 verás cómo se consigue esto.



```
20 CLS
30 A$=INKEY$:IF A$=""THEN 30
```



```
40 IF A$="d" THEN PRINT
   "ESTUPENDO":STOP
50 LOCATE14,10:PRINT"AUUG!"
```

La línea 40 comprueba si la tecla «d» ha sido pulsada; en otras palabras ¿es A$ igual a d? Si no es así, tu MSX ignorará la línea 40 y continuará en la 50 (trata de descubrir por qué es necesario el STOP). En este programa hay una cosa más de importancia. La «d» debe escribirse entre comillas. De otra forma el ordenador podría confundirla con una variable.

LANZAMIENTO DE UN MISIL

Vamos a utilizar lo que hemos aprendido en un programa que lanza un misil cuando se pulsa la tecla «f». Este es el listado:

```
20 CLS:KEY OFF
30 LOCATE 15,22:PRINT"*^*"
40 A$=INKEY$:IFA$=""THEN 40
50 IF A$<>"f" THEN 40
55 Y=21
60 LOCATE 16,Y:PRINT"^"
70 Y=Y-1
75 FOR J=1 TO 10:NEXT
80 LOCATE 16,Y+1:PRINT" "
90 IF Y>0 THEN 60
```

En la línea 20 se limpia la pantalla y se eliminan los códigos de las teclas de función (KEY OFF). La línea 30 dibuja la base de lanzamiento de misiles. Al llegar a la línea 40 el programa espera hasta que se pulsa una tecla. En la línea 50 se comprueba si la tecla pulsada es la «f». Si no es así, el programa vuelve a la línea 40 y queda esperando una nueva pulsación. En la línea 60 se dibuja el misil, que acaba de ser lanzado, en la fila Y de la pantalla. La primera vez Y vale 21 pero a medida que el misil asciende, Y se decrementa en una unidad. Esto tiene lugar en la línea 70. La línea 80 se encarga de borrar la posición que ocupaba anteriormente el misil. Por último, en la línea 90 se comprueba si el misil ha alcanzado la parte superior de la pantalla (en cuyo caso la coordenada Y vale 0) para terminar el programa.

MOVIENDOSE POR LA PANTALLA

El programa de la base de misiles, tal como está, es más bien aburrido, pero si se dota a la base de movimiento, las cosas mejoran un poco. Veamos cómo puede moverse la base.

```
20 P=15
30 CLS:KEY OFF
40 LOCATEP,22:PRINT"*^*"
50 A$=INKEY$:IF A$=""
   THEN 50
60 IF A$="i" THEN P=P-1:
   GOTO90
70 IF A$="d" THEN P=P+1:
   GOTO90
80 GOTO 50
90 IF P>33 THEN P=33
100 IF P<1 THEN P=1
110 GOTO30
```

La variable P de la línea 20 establece la posición inicial de la plataforma lanzamisiles, que queda dibujada en la pantalla al ejecutarse la línea 40. Entre las líneas 50, 60 y 70 se encuentra distribuida la rutina de lectura del teclado que comprueba si se ha pulsado la tecla I o la tecla D.

Al pulsar I, se resta uno de la variable P con lo que la base de misiles se desplaza a la izquierda. Pulsando D, la base se desplaza hacia la derecha al incrementarse en uno el valor de P.

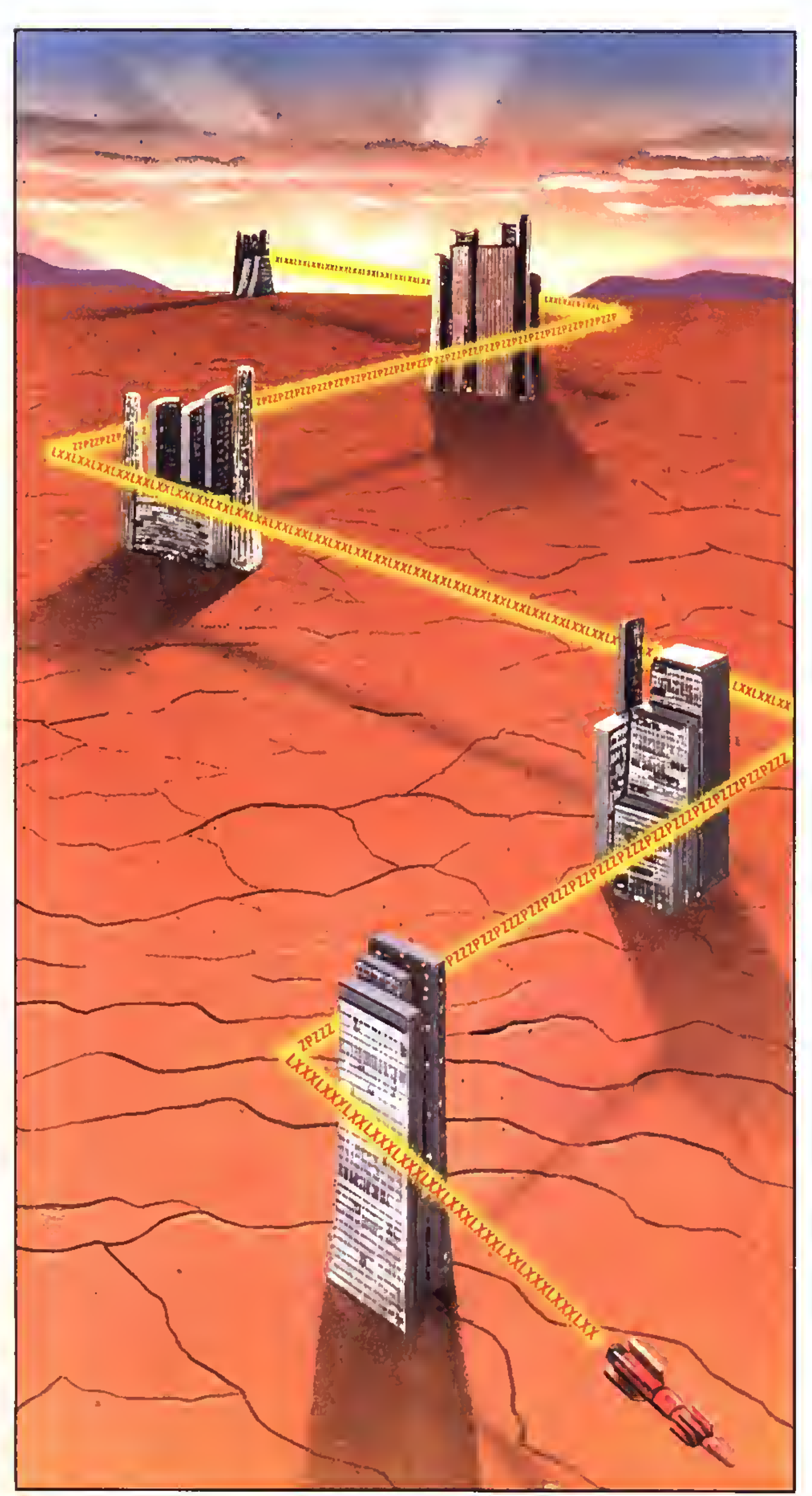
El GOTO de la línea 80 hace que el programa vuelva a la línea 50 si se pulsó cualquier otra tecla. Las líneas 90 y 100 limitan los valores de P para evitar que la base de misiles pueda salirse de la pantalla.

Finalmente la línea 110 hace que el programa vuelva a la línea 30 y pueda dibujarse la plataforma lanzamisiles en su nueva posición.

CREA TU PROPIO JUEGO

Ahora que ya dispones de algunos conocimientos y de algún bloque constructivo puedes acometer la realización de un juego sencillo que utilice estos bloques. Teclea el programa que sigue y escribe RUN.

```
10 KEY OFF:COLOR15,12,12
20 P=15
40 CLS
```



```
50 A=RND(-TIME)
60 A=INT(RND(0)*30)+3
70 LOCATEA,1:PRINT"*"
80 LOCATEP,22:PRINT" *^* "
90 A$=INKEY$:IF A$=""
   THEN 90
95 IF A$="z" THEN P=P-1
100 IF A$="x" THEN P=P+1
105 IF P>33 THEN P=33
110 IF P<1 THEN P=1
115 IF A$="f" THEN P1=P+2:
    D=21:GOTO130
120 GOTO 80
130 LOCATEP1,D:PRINT"^":
    D=D-1
140 FOR I=1 TO 10:NEXT
150 LOCATEP1,D+1
160 PRINT" "
170 IF D>1 THEN 130
180 IF P1=A THEN 40
200 GOTO 80
```

Verás aparecer una estrella cerca de la parte superior de la pantalla. Con las teclas Z e Y podrás trasladar la plataforma lanzamisiles hacia la derecha y hacia la izquierda hasta situarla bajo la estrella. Pulsa entonces la tecla F para lanzar un misil y destruir la estrella.

El programa está compuesto de tres secciones: hasta la línea 80, de la 90 a la 120, y las líneas 130 a 200 son análogas a las del anterior programa de lanzamiento de misiles. Se han cambiado las variables y el GOTO, pero lo único nuevo es la línea 180. En ella se comprueba si la estrella y el misil están en la misma vertical. Si ocurre esto el programa vuelve a empezar.

La sección central, líneas 90 a 120, es una versión resumida del movimiento por la pantalla que vimos anteriomente.

La primera sección del programa hasta la línea 80, realiza varias funciones. Las líneas 50, 60 y 70 generan un valor aleatorio que representa la posición de la estrella. Por su parte la línea 80 establece la posición de partida de la plataforma de misiles dibujándola a continuación.

Cuando hayas tecleado el programa escribe RUN y procura afinar tu puntería.

RUTINAS DE TANTEO Y TIEMPO

No hay nada como saber que sólo dispones de dos segundos para regresar a tu base, o que te faltan diez puntos para obtener la máxima puntuación en tu juego de batallas. Todos los juegos de marcianitos llevan algún tipo de cuenta de tanteo y tiempo, para darle más emoción. Con algunos programas sencillos, tú puedes hacer lo mismo.

Casi todos los juegos de ordenador necesitan algún tipo de puntuación o temporización, o incluso ambas cosas. Sin ellos no puedes juzgar lo bien que se te da el juego, o si vas mejorando algo, y no suele tener mucho interés jugarlo con tus amigos.

Podrías tener a alguien sentado tras de ti que fuera contando los impactos que consigues sobre tu feroz enemigo, pero esto no tiene mucho sentido cuando puedes programar a tu ordenador para que los cuente él. Con unas cuantas líneas de programa más, la máquina recordará también las puntuaciones.

Por la misma razón no hay necesidad de recurrir a un cronógrafo para la medida del tiempo. Todas las máquinas llevan un reloj incorporado, y puedes servirte de él de muchas formas para mejorar tus juegos.

CAMPO DE MINAS

Para que veas la manera de incorporar en la práctica las rutinas de puntuación y temporización, aquí tienes un juego en el que se van añadiendo las rutinas progresivamente. Cada rutina es muy sencilla y se puede añadir también a otros juegos.

El juego se llama Campo de Minas, y en él tú vas conduciendo un carro de combate, cuya misión es rescatar a unos paracaidistas que se han arrojado temerariamente sobre un campo minado. Cada vez que el tanque se



mueve, corre el riesgo de hacer detonar una mina plantada aleatoriamente por el ordenador. Como en un campo minado de verdad, las minas son invisibles, por lo que tendrás que moverte con precaución.

El tanque (desafortunadamente es sólo un signo #, hasta que aprendas la manera de combinar gráficos y movimiento en un programa BASIC) se controla utilizando las teclas de movimiento de cursor. Para ello se hace uso de la funcion STICK(0) del BASIC **MSX**, que se encarga de leer las teclas de cursor y de proporcionar uno u otro valor según la tecla que se haya pulsado.

De hecho, el nucleo del programa está constituido por la rutina de «moverse por la pantalla» que ya conoces.

Cuando teclees esta sección del juego y la ejecutes (con RUN), verás que todavía no está completa: después de que hayas rescatado al paracaidista, no sucede nada, excepto que el tanque continúa vagando sin rumbo por la pantalla. El programa ha de detenerse pulsando la tecla CTRL/STOP o tendrás que esperar hasta que el tanque tropiece con una mina escondida. Pero no te alarmes, todo esto mejorará en cuanto le pongas las rutinas de puntuación y de tiempo que siguen.

```
5 TX=16:TY=5
10 WIDTH 40
12 A=RND(-TIME)
15 COLOR 15,4
80 CLS:LOCATE 0,12:PRINT "---
   ---------------------------
   -----------"
90 PX=INT(RND(1)*30)+1
100 PY=INT(RND(1)*10)
110 IF PX=TX AND PY=TY THEN
    GOTO 90
120 LOCATE PX,PY:PRINT"O"
    :LOCATE TX,TY:PRINT"#"
130 AX=TX:AY=TY
140 A=STICK(0)
145 IF A=1 THEN TY=TY-1
150 IF A=5 THEN TY=TY+1
160 IF A=7 THEN TX=TX-1
170 IF A=3 THEN TX=TX+1
190 IF TY<0 OR TY>11 THEN
    TY=AY
200 IF TX<0 OR TX>39 THEN
    TX=AX
230 LOCATE AX,AY:PRINT" "
240 LOCATE TX,TY:PRINT"#"
250 MX=INT(RND(1)*30)+1
260 MY=INT(RND(1)*10)
270 IF MX=TX AND MY=TY THEN
    LOCATE MX,MY:PRINT" "
    :LOCATE 0,14:PRINT
    "BOOM!!-TE ALCANZO UNA
    MINA":STOP
310 GOTO 130
```

El programa comienza dividiendo la pantalla en dos mitades, mediante una línea de trazos que se dibuja desde la línea 80 del programa. Previamente, entre las líneas 5 y 15 se ha definido el color y la anchura de la pantalla así como la posición inicial del tanque en las variables TX y TY.

En las líneas 90 y 100 se eligen aleatoriamente las coordenadas del punto de caída del paracaidista. Esta posición se compara con la que ocupa el tanque en la línea 110. Si son iguales, se elige una nueva posición para el paracaidista. Tanto éste como el tanque se dibujan en la pantalla al ejecutarse la línea 120.

Entre las líneas 140 y 170 nos encontramos con la rutina de lectura de las teclas de cursor.

Para evitar que el tanque pueda salirse de los límites de la pantalla se utilizan las líneas 190 y 200.

En las líneas 250 y 260 se elige aleatoriamente una posición para la mina. A continuación, en la línea siguiente, se comparan las posiciones de la mina y del tanque. Si ambas coinciden se produce la explosión y aparece en la pantalla la palabra BOOM.

La línea 310 devuelve el programa al principio del bucle de lectura de teclado.



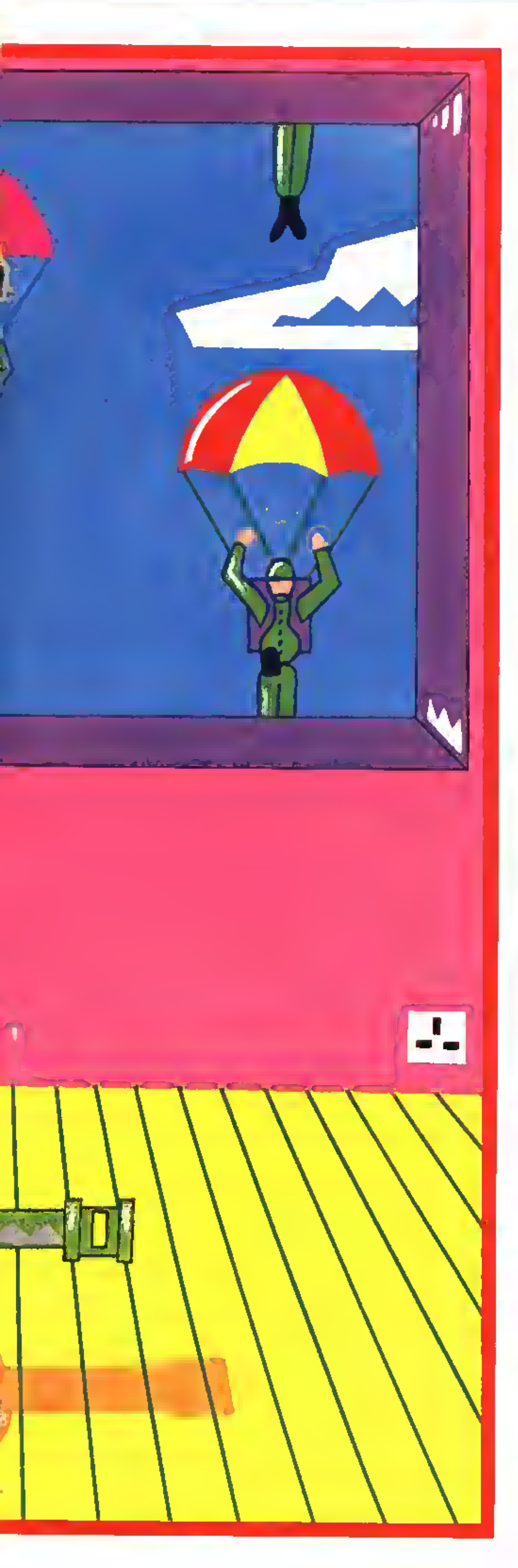
PUNTUACION

En los juegos de extraterrestres, la puntuación aumenta normalmente cuando la posición ocupada por dos objetos en la pantalla es la misma.

Los objetos pueden ser un misil y un blanco, un comecocos y una píldora alimenticia, un tanque y un paracaidista, o lo que el juego requiera.

Añade, pues, estas líneas a tu programa para ver cómo trabaja en la práctica el mecanismo de tanteo.

```
40 P=0
280 IF PX=TX AND PY=TY THEN
```



```
    P=P+1:GOTO80
330 LOCATE 0,16:PRINT P;
    "Paracaidistas recogidos"
```

Cambia el STOP de la línea 270 por un GOTO 330. La línea 270 se convierte ahora en:

```
270 IF MX=TX AND MY=TY THEN
    LOCATE MX,MY:PRINT" "
    :LOCATE 0,14:PRINT
    "BOOM!!-TE ALCANZO UNA
    MINA":GOTO330
```

La línea 280 es importante. En ella se comprueba que el tanque y el paracaidista ocupan la misma posición en la pantalla. Si ocurre esto, la puntuación aumenta en 1.

En la línea 40 se pone a cero el tanteo antes de que empiece el juego y la línea 330 visualiza la puntuación. Cambiando el STOP de la línea 270 se consigue que el ordenador presente la puntuación después de haber tocado una mina.

El jugador se enfrenta ahora con una sucesión de paracaidistas a los que rescatar. Cada vez que es rescatado uno, cae otro del cielo. El juego se detiene cuando se produce la explosión de una mina (porque el tanque ocupa la misma posición en la pantalla).

PUNTUACION MAXIMA

No es difícil añadirle a tu juego una opción de puntuación máxima. No tienes más que introducir una variable asociada a esta puntuación —por ejemplo, PM— y algún método para actualizarla cuando sea superada dicha opción, además hay que poner una rutina de presentación.

Aquí tienes las líneas que debes añadir para tener una opción de puntuación máxima.

```
30 PM=0
350 IF P>PM THEN PM=P
370 LOCATE0,17:PRINT"Mayor
    puntuacion";PM
```

En primer lugar debes poner la puntuación máxima en su menor valor posible, por lo que la línea 30 pone PM a cero. Después de que el juego se ha detenido, la línea 350 compara la última puntuación (P) con la puntuación máxima (PM). Si la puntuación obtenida es mayor que la puntuación máxima se actualiza PM, haciéndola igual a P. Finalmente, la línea 370 visualiza en la pantalla el valor.

Es probable que estas líneas te parezcan suficientes para dotar el juego de un tanteo. Por desgracia, esto no es cierto. Cada vez que hagas ejecutar el programa (con RUN), el ordenador olvida automáticamente el valor de PM, y los valores de las otras variables. Para mantener el valor de PM tienes que añadir las líneas de «¿Otra vez?» que se describieron en un capítulo anterior.

```
390 FOR F=1 TO 1000:NEXT F
410 LOCATE 0,19:PRINT"Otra
    vez ? (S/N)"
420 A$=INKEY$:IF A$="" THEN
    420
430 IF A$="s" THEN GOTO 40
440 IF A$="n" THEN CLS:END
450 GOTO 420
```

He aquí lo que hacen estas nuevas líneas:

Hay un corto retardo introducido por la sentencia FOR ... NEXT en la línea 390. El mensaje «Otra vez» (S/N) se presenta en la línea 410.

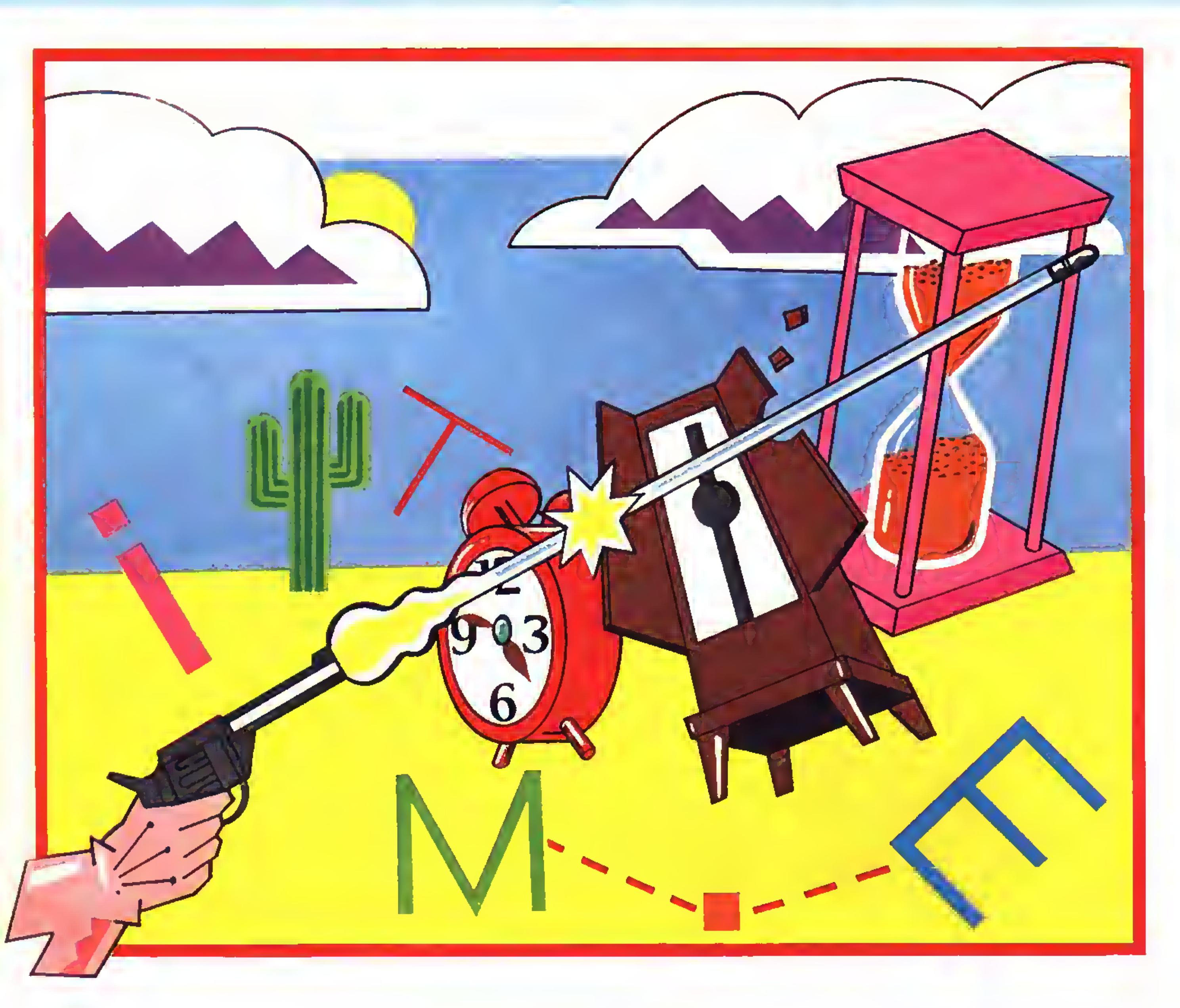
La rutina «Otra vez» está en las líneas 410 a 450. La línea 430 hace que el programa reeomience en la línea 40 si se pulsa S, y la línea 440 detiene el programa si se pulsa N. La línea 450 sirve para asegurarse de que cualquier otra tecla será ignorada.

Como no hay necesidad de pulsar RUN cada vez que desees jugar de nuevo, el valor de PM será preservado, aunque al volver a cargar el programa (con LOAD) se perderá el valor de PM, incluso si encuentras alguna forma de arrancar el programa sin pulsar RUN.

MEDIDA DE TIEMPOS

Tal como está el juego, depende demasiado de la suerte, simplemente, el jugador sigue adelante hasta que pisa una mina escondida.

Se puede introducir en este tipo de juegos un elemento de habilidad, para convertirlo en una carrera contra el reloj. Con los siguientes añadidos puedes cronometrar cuánto tardas en rescatar a diez paracaidistas.




```
75 TIME=0
290 IF P<10 AND TX=PX AND
    TY=PY THEN GOTO 80
300 IF P=10 THEN GOTO 320
320 T=TIME/50
340 IF P=10 THEN LOCATE 0,20
    :PRINT T;"Segundos"
```

El reloj interno de la máquina está corriendo todo el tiempo que el ordenador permanece encendido. Para arrancar el contador de tiempos, tienes que poner a cero la lectura de dicho reloj. Esto se hace en la línea 75. Simplemente teclea TIME=0

El reloj se «detiene» en la línea 320. Realmente, el reloj no puede detenerse; lo que tú haces es que la máquina recuerde una lectura particular en un instante determinado, por ejemplo cuando coinciden dos objetos en la pantalla.

El reloj está constituido por la variable TIME, que se encarga de llevar la cuenta del tiempo, y que se incrementa en una unidad, 50 veces por segundo.

TIME no para de incrementarse hasta que llega a 65535. Entonces vuelve a cero y empieza a contar otra vez.

Si nosotros escribimos TIME=0, como en la línea 75, la variable se pone a cero y empieza a contar. Al leer posteriormente la varible TIME, como hace la línea 320, el valor que leemos es el tiempo que ha transcurrido desde que pusimos a cero dicha variable. Así es como llevan la cuenta del tiempo la mayoría de los programas.

El reloj debe detenerse cuando el jugador ha rescatado a diez paracaidistas, por lo que en la línea 300 se comprueba si ya se han rescatado diez, en cuyo caso, el programa salta a la línea 320, que es la que «detiene» al reloj. La línea 340 imprime el tiempo que se ha tardado en rescatar a diez paracaidistas.

Si se ha rescatado con éxito a un paracaidista y, **además**, el número total de los que van ya rescatados es menor que diez, la línea 290 hace caer otro.

La línea 340 presenta el tiempo transcurrido para el rescate sólo si se han rescatado ya los diez. La lectura de tiempo es dividida por 50, por lo que el tiempo aparece en segundos. El reloj es actualizado 50 veces cada segundo.

MEJOR TIEMPO

De la misma forma que añadiste antes una opción de tanteo máximo para el juego, puede resultarte interesante una opción de mejor tiempo. En esta variante del juego tienes que poder registrar el tiempo más rápido en que los diez paracaidistas son rescatados, aunque este principio se puede aplicar a cualquier temporización que desees hacer.

He aquí las líneas que tienes que añadir:

```
20 LT=999999!
360 IF T<LT AND P=10
    THEN LT=T
380 LOCATE 0,21:PRINT
    "Mejor tiempo";LT
```

Igual que con el tanteo máximo se ponía inicialmente un «tanteo máximo» muy bajo, ahora se pone un «tiempo récord» ridículamente largo.

La línea 20 asigna a la variable de mejor tiempo (LT) un valor de 999999.

La línea 360 compara el último tiempo obtenido con el mejor tiempo. Si el último tiempo obtenido es más corto que el tiempo récord y, **además**, se han rescatado ya diez paracaidistas, entonces se modifica el tiempo récord haciéndolo igual al último tiempo obtenido.

Finalmente, la línea 380 sirve para expresar en segundos el tiempo récord. La variable de tiempo récord viene dividida por 50 para que resulte en segundos.

Una cosa que has de recordar es que si estás utilizando una opción de tiempo récord en un juego, tienes que utilizar la rutina de «¿Otra vez?», pues, de lo contrario, el valor del tiempo récord se perderá cada vez que ejecutes (con RUN) el programa.

EL TECLADO Y LA CUENTA DEL TIEMPO

Hasta ahora has visto cómo puedes controlar el reloj interno de la máquina desde dentro de un programa, examinando las posiciones de dos objetos sobre la pantalla. Otra forma de «detener» el reloj es servirte del teclado de tu ordenador.

Puedes hacerlo con la sentencia INKEY$. Resulta tan fácil como arrancar y parar un cronómetro para controlar el movimiento de los objetos por la pantalla.

Aquí tienes un juego de acción rápida que ilustra cómo puede usarse el teclado para detener el reloj:

```
20 CLS
30 A=RND(-TIME):N=INT
   (RND(1)*900)+1
40 FOR F=0 TO N
50 NEXT F
55 FOR F=1 TO 50:C$=INKEY$
   :NEXT F
60 CLS:LOCATE 0,10:PRINT
   "Dispara!!"
70 TIME=0
80 A$=INKEY$:IF A$="" THEN
   80
90 T=TIME
100 LOCATE 0,10:PRINT
    "BANG!!!!!"
110 FOR F=1 TO 300
120 NEXT F
130 M=INT(RND(1)*35)+1
140 IF T<M THEN LOCATE 0,15
    :PRINT"Has sobrevivido"
150 IF T>M THEN LOCATE 0,15
    :PRINT"Has muerto      "
160 IF T=M THEN LOCATE 0,15
    :PRINT"Habeis muerto   "
```

El programa presenta el mensaje «DISPARA!!» y el jugador ha de pulsar cualquier tecla tan rápido como pueda. Se mide el tiempo de reacción desde el momento en que apareció el mensaje.

Las líneas 30 a 50 introducen una pausa aleatoria. La línea 60 sirve para enviar el mensaje «DISPARA!!» e inmediatamente se arranca el contador de tiempo en la línea 70. La línea 80 hace que la máquina espere, continuando cuando se ha pulsado una tecla cualquiera. Ya vimos esta línea al ocuparnos del «Control del Teclado».

En cuanto se ha pulsado una tecla cualquiera, la línea 90 para el contador, llamado T a la lectura que tiene en ese momento. La línea 100 escribe «BANG!!». Hay una pausa introducida por las líneas 110 y 120 antes de que la máquina elija un instante de tiro. La línea 130 es la que se encarga de hacer esto.

La máquina tiene ahora dos variables, su tiempo, T, y el tiempo de la máquina, M. Las líneas 140 a 160 comparan estos valores y presentan el resultado del duelo.

P y R

¿Existe algún límite para la duración máxima que se puede tener?

Sí existe un límite, aunque normalmente es tan alto que en la práctica no tiene importancia. El reloj interno de casi todos los ordenadores domésticos avanza a la misma velocidad, y el factor limitador es la cantidad de pulsos de tiempo que el ordenador puede recordar.

En tu **MSX** se puede contar hasta dos bytes (65535) lo que equivale aproximadamente a unos 22 minutos.

Foto de la pantalla de un momento del juego.



JUEGOS DE LABERINTO

- LOS PRINCIPIOS DE LA ANIMACION
- MOVIMIENTO DE GRAFICOS
- COMO UTILIZAR LOS GRAFICOS INCORPORADOS

Los juegos de laberintos sofisticados requieren programas largos. Pero tú puedes diseñar algunos sencillos extrayendo de ellos importantes principios y utilizando poco más que un bucle y sentencias DATA.

Los juegos de laberintos ejercen una fascinación permanente sobre los propietarios de un ordenador, por lo que las casas de *software* continúan sacando nuevas variantes del comecocos.

Este artículo te enseñará la manera de saltar al carro de los fabricantes de laberintos, permitiendo que te construyas el tuyo propio.

En su primera fase el laberinto no incluye «enemigos» ni obstáculos, ya que esto requeriría un programa muy largo. Pero te enseñará cómo se programa el que tu carácter principal no pueda atravesar las paredes, lo cual es la base de todos los juegos de esta clase. También se incluye la puntuación y el crono, así como una rutina de «mejor tanteo», a fin de darle un cierto interés competitivo.

La manera más fácil de entender cómo funciona el juego del laberinto es ir introduciéndolo por etapas. Así pues, empieza construyendo el propio laberinto:

```
100 CLS:FOR N=3 TO 17
110 READ A$
120 FOR M=7 TO 21
130 LOCATE M,N:PRINT"."
140 IF MID$(A$,M-6,1)="p"
    THEN LOCATE M,N:PRINT
    CHR$(219)
150 NEXT M
160 NEXT N
9000 DATA "ppppppppppppppp"
9010 DATA "p.............p"
9020 DATA "p.pp.pp.pp.pp.p"
9030 DATA "p.p.........p.p"
9040 DATA "p...p.p.p.p...p"
9050 DATA "p.ppp.p.p.ppp.p"
9060 DATA "p.....p.p.....p"
9070 DATA "pppp.pp.pp.pppp"
9080 DATA "p.....p.p.....p"
9090 DATA "p.ppp.p.p.ppp.p"
9100 DATA "p...p.p.p.p...p"
9110 DATA "p.p.........p.p"
9120 DATA "p.pp.pp.pp.pp.p"
9130 DATA "p.............p"
9140 DATA "ppppppppppppppp"
```

Las líneas 100, 120, 150 y 160, que definen un par de bucles FOR ... NEXT, establecen los contornos del laberinto. Con la línea 130 se imprime un punto en cada cuadrado.

Las líneas 110 y 140 se encargan de leer los datos de las líneas 9000 a 9140 y de sustituir el punto por un bloque macizo, que es el carácter que corresponde al código **ASCII** 219, cada vez que en la sentencia DATA correspondiente aparece la letra p. De esta forma se obtiene un laberinto de paredes macizas en el que los pasillos interiores están cubiertos de puntos. Como podrás ver cuando teclees RUN se trata del clásico laberinto de los programas del tipo **comecocos**.

CONSTRUYENDO EL «COMILON»

Sin embargo, un laberinto resulta bastante inútil si no hay algo que se mueva a través de él. Ejecuta pues (con RUN) el programa, y añádele lo siguiente:

```
40 WIDTH 40
50 X=14
60 Y=10
```





```
1000 LOCATE X,Y:PRINT "*"
1010 XX=X
1020 YY=Y
1030 B=STICK(0):IF B=0
     THEN 1030
1040 IF B=1 AND VPEEK(40*
     (Y-1)+X)<>219 THEN Y=Y-1

1050 IF B=5 AND VPEEK(40*
     (Y+1)+X)<>219 THEN Y=Y+1

1060 IF B=3 AND VPEEK
     (40*Y+X+1)<>219 THEN
     X=X+1
1070 IF B=7 AND VPEEK
     (40*Y+X-1)<>219 THEN
     X=X-1
1080 LOCATE XX,YY:PRINT" "
1090 GOTO 1000
```

Como ya has experimentado con el control de movimientos mediante las teclas de cursor, la mayor parte de estas nuevas líneas te resultarán familiares. Lo que hacen es colocar un **comecocos** (en este caso se trata de un sencillo asterisco) en el interior del laberinto, haciendo que se mueva en la dirección que tú le indiques mediante las teclas de cursor.

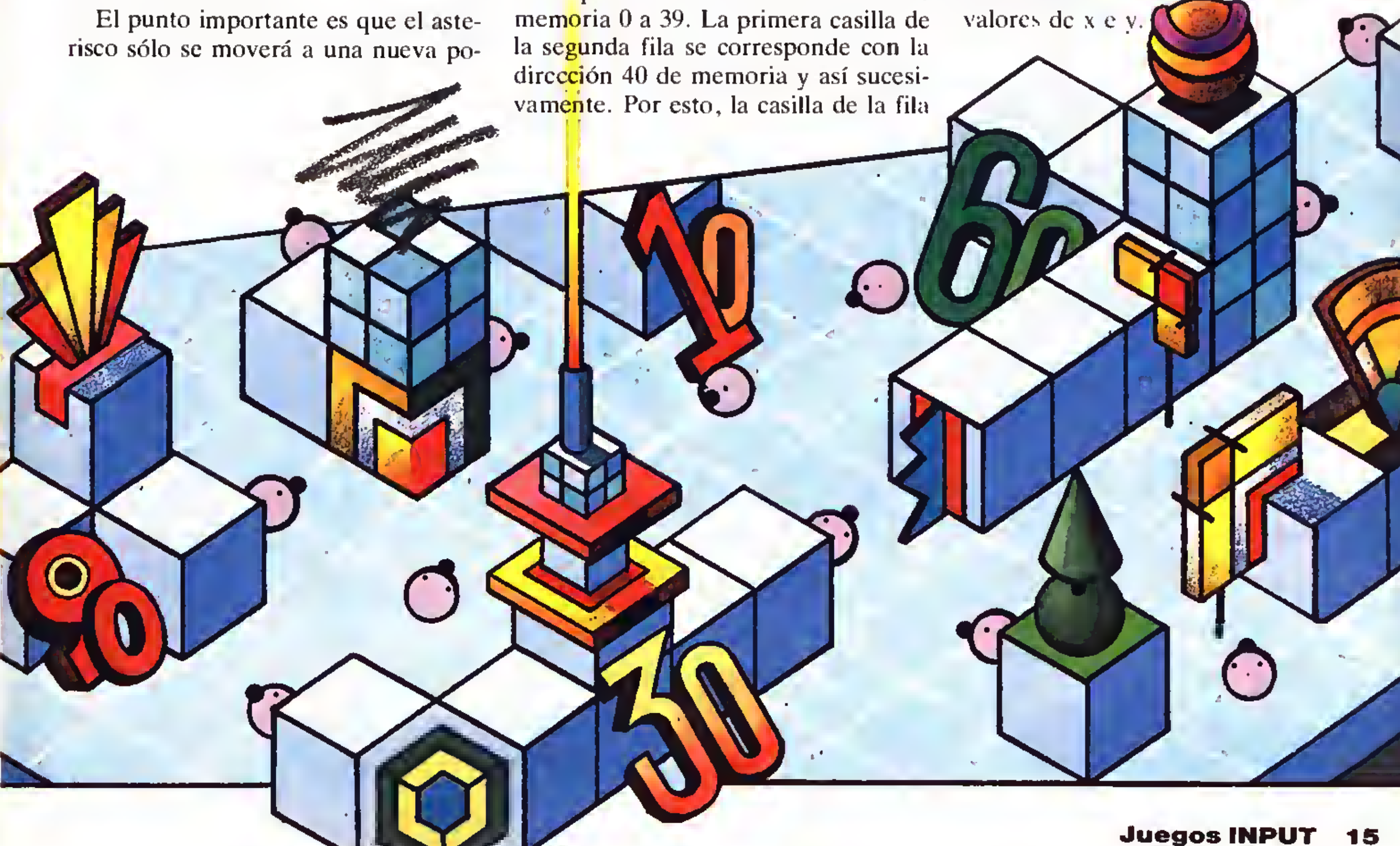
El punto importante es que el asterisco sólo se moverá a una nueva posición si el muro del laberinto se lo permite. Para que lo entiendas, imagina que el asterisco está situado en el laberinto en las coordenadas x,y. Si quieres desplazarlo una casilla hacia arriba tendrás que restarle uno a la coordenada y, situando el asterisco en las nuevas coordenadas x, y−1. Pero esto sólo podrás hacerlo si en estas nuevas coordenadas no hay muro. Para saber si esto ocurre, el programa utiliza la instrucción VPEEK, que permite leer directamente los contenidos de la memoria de pantalla (la famosa **VRAM**).

En próximos números de **INPUT** comentaremos más a fondo esta instrucción y la forma de manejar con ella la **VRAM**, pero por el momento nos basta con saber que VPEEK(40*y+x) proporciona el valor **ASCII** del caracter que se encuentra en las coordenadas x,y de la pantalla. Esto es así porque la memoria de pantalla se organiza en direcciones consecutivas de memoria que representan las casillas de la pantalla. Como estamos trabajando en 40 columnas, la primera fila de la pantalla se corresponde con las direcciones de memoria 0 a 39. La primera casilla de la segunda fila se corresponde con la dirección 40 de memoria y así sucesivamente. Por esto, la casilla de la fila y, columna x, tiene como dirección de memoria 40*y+x.

Entre las líneas 1030 y 1070 el programa se encarga de ver si se ha pulsado alguna tecla de cursor. Si ha sido así, mueve el asterisco a la nueva posición, pero sólo tras comprobar que el caracter que ocupa dicha posición no tiene el código **ASCII** 219, es decir, no es el muro.

Por su parte, las líneas 1010, 1020 y 1080 definen la posición que el asterisco acaba de abandonar y se encargan de borrar los puntos comidos, poniendo un espacio en blanco mediante PRINT. Estas líneas son muy importantes en cualquier programa de este tipo y se encargan de algo que, aunque parece trivial, hay que tener siempre muy en cuenta.

Se trata de que al cambiar el asterisco de la posición x,y a una nueva posición, que también se llama x,y, hay que guardar en algún sitio los anteriores valores x,y, para que el programa sepa donde estaba antes el asterisco y pueda borrarlo. Para ello este programa utiliza las variables xx,yy en las que se guardan temporalmente los anteriores valores de x e y.



PUNTUACION Y CRONOMETRAJE

Ya tienes todos los elementos para construir un juego sencillo. Para que de verdad se pueda jugar con él, y en ausencia de «enemigos» —que harían el programa desmedidamente largo— lo mejor es incluir una rutina de cronometraje y tanteo. Agrega pues las siguientes líneas:

```
10 BT=100000!
20 PUN=0
30 TIME=0
1085 IF VPEEK(40*Y+X)=46
     THEN PUN=PUN+1
1087 LOCATE 0,20:PRINT
     "Puntos ";PUN
1090 IF PUN=110 THEN
     T=TIME:GOTO 2000
1095 GOTO 1000
2000 LOCATE X,Y:PRINT" "
2010 LOCATE 0,20:PRINT
     "Tiempo empleado:";
     T/50;"Segundos"
2020 IF T<BT THEN BT=T
2030 LOCATE0,21:PRINT
     "Mejor tiempo";BT/50
```

El sistema de puntuación es bastante sencillo. La línea 20 pone a cero la puntuación al comenzar la partida. La línea 1085 incrementa en uno la puntuación cada vez que el asterisco se come un punto. Para ello se comprueba, mediante VPEEK, si en la casilla en que se coloca el asterisco había un punto (el código **ASCII** del punto es 46) o no lo había.

La línea 1087 se encarga de imprimir la puntuación en la parte inferior de la pantalla.

Cuando el asterisco se ha comido todos los puntos la puntuación vale 110. Entonces, la línea 1090 lee el tiempo que ha transcurrido y salta a la sección de cronometraje, en la línea 2000.

Esta sección es también muy fácil de entender. Empieza en la línea 10, fijando el «mejor tiempo» inicial en 100000, mucho más de lo que hará cualquier jugador.

A continuación, en la línea 30, se pone el reloj a cero con lo que da comienzo la cuenta del tiempo. Este transcurre segundo a segundo, hasta que el jugador se «come» todos los puntos del laberinto y consigue con ello que la puntuación llegue a 110. En ese momento y en la línea 1090 se lee el tiempo transcurrido. Entnces se pasa a la línea 2000, que se encarga de borrar al asterisco, y luego a la 2010 que imprime el tiempo empleado, en segundos. Como el reloj cuenta cincuentaavos de segundo, se divide por cincuenta y se obtienen segundos.

Por último, las líneas 2020 y 2030 comparan el tiempo obtenido con el «mejor tiempo». Si el tiempo obtenido es mejor, queda registrado como nuevo tiempo. En la línea 2030 se imprime este mejor tiempo, en segundos.

OTRA VEZ

Para darle al jugador otra oportunidad, tienes que introducir las siguientes líneas:

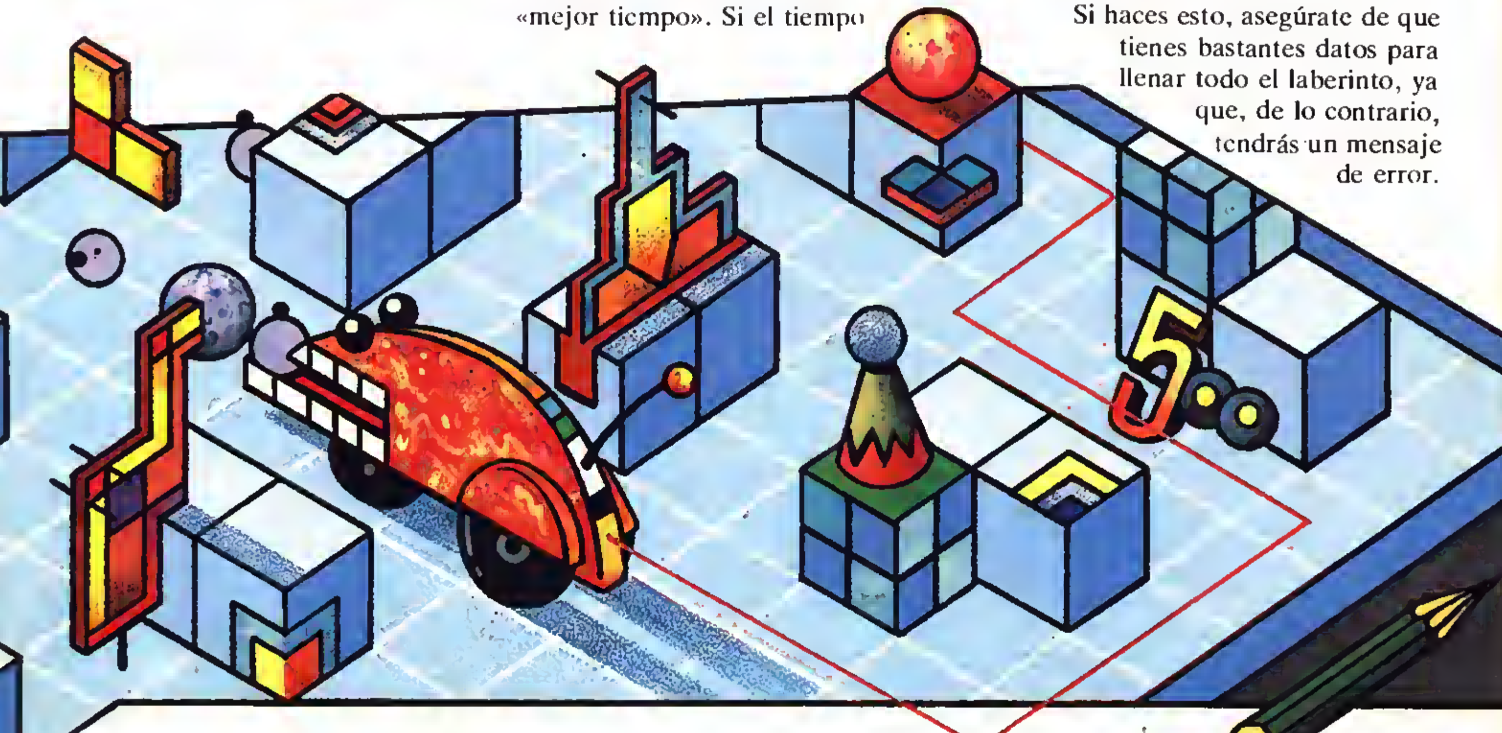
```
2040 FOR J=1 TO 1000:NEXT
2050 LOCATE0,22:PRINT
     "Otra vez ? (S/N)"
2060 C$=INKEY$:IF C$=""
     THEN 2060
2070 IF C$="s" THEN RESTORE
     :GOTO 20
2080 IF C$="n" THEN CLS:END
2090 GOTO 2060
```

Con RESTORE se vuelve al principio de la lista de los DATA.

OTROS LABERINTOS

Si quieres probar otros laberintos con las mismas dimensiones que éste, puedes hacerlo simplemente cambiando la disposición de las letras p en las sentencias DATA, desde la línea 9000 en adelante. Para tener un laberinto más grande o uno más pequeño, también tienes que volver a definir los límites o contornos, cambiando los números que aparecen en las líneas 100 y 120.

Si haces esto, asegúrate de que tienes bastantes datos para llenar todo el laberinto, ya que, de lo contrario, tendrás un mensaje de error.



SPRITES PARA TUS JUEGOS

- COMO CONSTRUIR SPRITES
- MODOS DE PANTALLA
- DEFINICION DE UN SPRITE
- COMO DAR MOVIMIENTO A LOS SPRITES

Los *sprites* son gráficos que tu mismo puedes definir y que te ofrecen unas enormes posibilidades a la hora de programar todo tipo de juegos, desde los más sencillos a los más complicados. Te vamos a enseñar como construir y manejar *sprites* con tu MSX para que puedas incorporar a estos simpáticos «duendes» en cualquiera de tus programas.

Los *sprites* son objetos gráficos constituidos por una agrupación de *pixels* (puntos de pantalla). Estos puntos, que forman parte de un rejilla cuadrada, puedes seleccionarlos a voluntad a la hora de la definición del *sprite*.

El primer paso para definir el *sprite* consiste en dibujarlo sobre un papel cuadriculado. Previamente tendrás que decidir el tamaño que va a tener tu *sprite*, de los cuatro tamaños posibles que te ofrece tu **MSX**. Este tamaño se fija mediante la instrucción SCREEN, cuyo formato es:

```
SCREEN [modo de pantalla],
[tamaño de sprites]
```

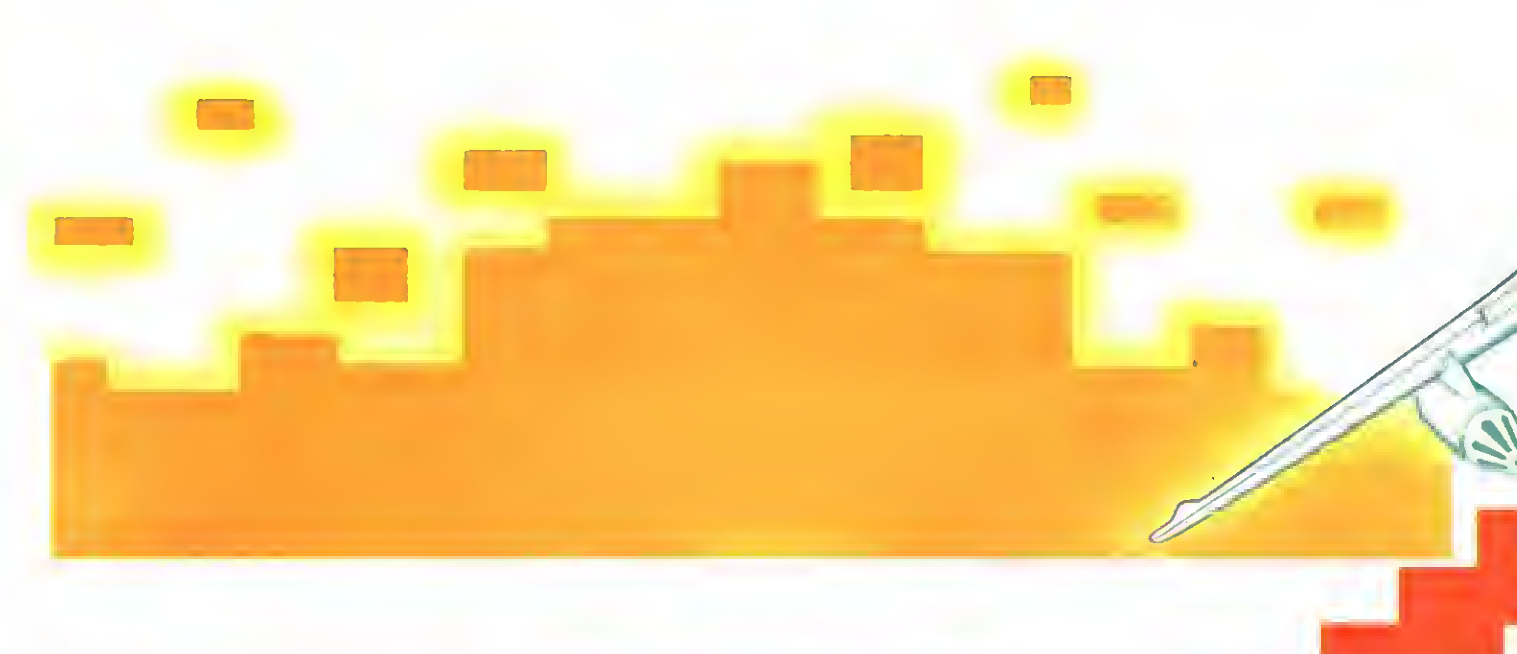
Hay 4 modos de pantalla en tu **MSX** que corresponden a los números 0, 1, 2 y 3. El modo 0 es un modo de texto en el que no puedes utilizar *sprites*. En cambio si puedes utilizarlos en los modos restantes.

Nosotros vamos a enseñarte el manejo de los *sprites* en el modo 2 que es el modo gráfico de alta resolución. En este modo puedes considerar la



pantalla como una rejilla formada por muchos puntos diminutos.

Hay 256 puntos horizontales por 192 puntos verticales, lo que nos proporciona una pantalla con 49152 puntos en total.



Trabajando en este modo de pantalla (SCREEN 2) puedes elegir entre cuatro tamaños diferentes para tus *sprites*: 8×8, 8×8 (expandido), 16×16 y 16×16 (expandido). En el primer caso tu *sprite* podrá ser cualquier agrupación de 64 puntos (8 en horizontal × 8 en vertical), en el segundo caso tendrás el mismo número de puntos pero estos serán el doble de grandes. Si optas por el tamaño 16×16, tu *sprite* podrá ser cualquier agrupación de 256 puntos y si escoges este mismo tamaño expandido, seguirás teniendo una rejilla de 256 puntos para dibujar tu *sprite*, pero, al igual que antes, los puntos serán de tamaño doble. Cada uno de los tamaños de *sprites* se define mediante uno de los números 0, 1, 2 y 3. Así por ejemplo:

```
SCREEN 2,2
```

te coloca en el modo de pantalla 2 y con *sprites* de 16×16 puntos, mientras que

```
SCREEN 2,1
```

corresponde al modo de pantalla 2, pero los *sprites* en este caso serán de 8×8 puntos a doble tamaño, es decir, expandidos.

DEFINICION DEL SPRITE

Vamos a definir y a introducir en el ordenador un *sprite* de 16×16 puntos; para ello lo primero es coger un papel cuadriculado y delimitar una rejilla de 16×16 como el de la figura. Sobre esta rejilla dibujamos nuestro *sprite* teniendo en cuenta que los puntos que llenemos de color se verán, mientras que los que dejemos en blanco permanecerán invisibles. Vamos a dibujar por ejemplo un marciano, como el que puedes ver en la figura que se muestra en la página 24.

El siguiente paso consiste en convertir el dibujo en valores númericos para introducirlos en el ordenador. Para ello tienes que dividir el *sprite* en

4 bloques de 8×8 puntos e introducir los datos de cada bloque (1, 2, 3 y 4) unos a continuación de otros y en el orden señalado. Empezando por el bloque 1, tienes que coger cada fila y convertirla en un número binario de 8



bits. Si un punto está coloreado, pones un 1 en el bit correspondiente. En caso contrario pones un cero. Cuando hayas escrito el número binario, puedes pasarlo a decimal utilizando la instrucción:

```
PRINT VAL(&B numero binario)
```

En nuestro ejemplo, y para el primer bloque obtenemos:

```
Bloque 1.    0000 0000     0
             0001 1000    24
             0000 0100     4
             0000 0111     7
             0000 0101     5
             0000 0111     7
             0000 0111     7
             0000 0001     1
```

Haciendo lo mismo para los bloques 2, 3 y 4 obtenemos una serie de 32 valores decimales. Estos valores definen al *sprite*. Vamos a introducirlos en el ordenador mediante el siguiente programa.

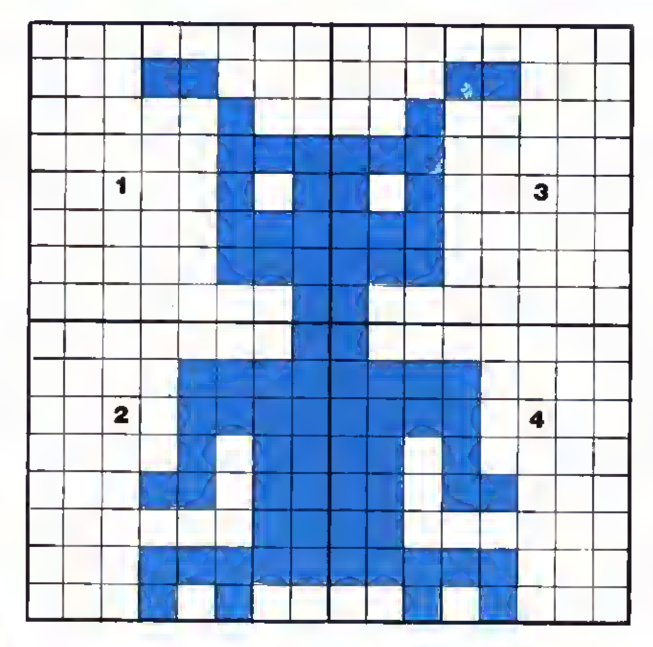
```
10 SCREEN2,2:CLS
15 FOR J=1 TO 32
20 READ A
25 S$=S$+CHR$(A)
30 NEXT J
35 SPRITE$(0)=S$
40 PUTSPRITE0,(100,100),15,0
45 GOTO45
1000 DATA 0,24,4,7,5,7,7,1
1010 DATA 1,15,15,11,27,1,31,
     26
1020 DATA 0,24,32,224,160,224
     ,224,128
1030 DATA 128,240,240,208,216
     ,192,248,40
```

Cuando ejecutes el programa (RUN) verás como al cabo de un momento aparece nuestro marciano en el centro de la pantalla.

El programa para el modo SCREEN 2 en la línea 10 define además el tamaño de 16×16 para el *sprite*. Entre las líneas 15 y 35 se leen los DATAs del *sprite* (líneas 1000 a 1020) y se asignan primero a la variable intermedia S$ y luego al *sprite* 0.

La línea 40 se encarga de colocar el *sprite* 0 en el plano 0, en las coordenadas x=100, y=100 y en color blanco (15).

Por último la línea 45 es un bucle infinito para evitar el salirnos del modo SCREEN 2.



El marcianito de la figura se puede realizar mediante un sprite *de 16×16 puntos formado por 4 bloques de 64 puntos. Cada punto corresponde a un bit que tendrá el valor 1 donde hay dibujo y 0 donde no lo hay.*

MUEVE TUS SPRITES

La instrucción PUTSPRITE, sirve para colocar los *sprites* sobre la pantalla, pero también para moverlos. Cuando escribimos, por ejemplo:

PUTSPRITE 0,(x,y),15,0

estamos colocando en el plano 0 (hay 32 planos de *sprite*) y en las coordenadas (x,y) a nuestro *sprite* 0. Si hacemos variar las coordenadas x,y, tenemos que el *sprite* se mueve. Además, —y esta es una característica de los *sprites*— cuando lo colocamos en una nueva posición, dentro de un plano de *sprite*, no hace falta que borremos al *sprite* de su posición anterior, de ello se encarga el *chip* de vídeo, ahorrándonos trabajo y tiempo.



Añade las líneas siguientes al programa para ver cómo tu *sprite* se mueve aleatoriamente por la pantalla.

```
45 FOR J=1 TO 1000:NEXT J
47 XX=100:YY=100
50 R=RND(-TIME)
55 L=RND(1)*50
65 DX=1:IF RND(1)>.5 THEN DX=
   -1
70 DY=1:IF RND(1)>.5 THEN DY=
   -1
80 FOR J=1 TO L
82 IF(XX+DX)>255 OR(XX+DX)<0
   THEN DX=0
83 IF (YY+DY)>192 OR (YY+DY)<
   0 THEN DY=0
85 PUTSPRITE0,(XX+DX,YY+DY),
   15,0
86 XX=XX+DX:YY=YY+DY
90 NEXT J
100 GOTO55
```

El programa funciona de la siguiente forma: La línea 45 es un bucle de espera. En la línea 47 se almacenan en xx e yy las coordenadas de la posición actual del *sprite*. Estas coordenadas van a ir variando lo que va a producir el movimiento del *sprite*. En las líneas 50 y 55 se define una longitud aleatoria (L) para el camino que va a recorrer el *sprite* cada vez que se mueva. Las líneas 65 y 70 definen, también aleatoriamente, la dirección en la que se va a mover el *sprite*. Esta puede ser cualquiera de las 4 direcciones diagonales. Por ejemplo si DX=1 y DY=1, se incrementarán positivamente las coordenadas x e y, con lo que el *sprite* se moverá hacia abajo y a la derecha.

Si resultan DX=−1 y DY=1, el movimiento será hacia arriba y a la derecha.

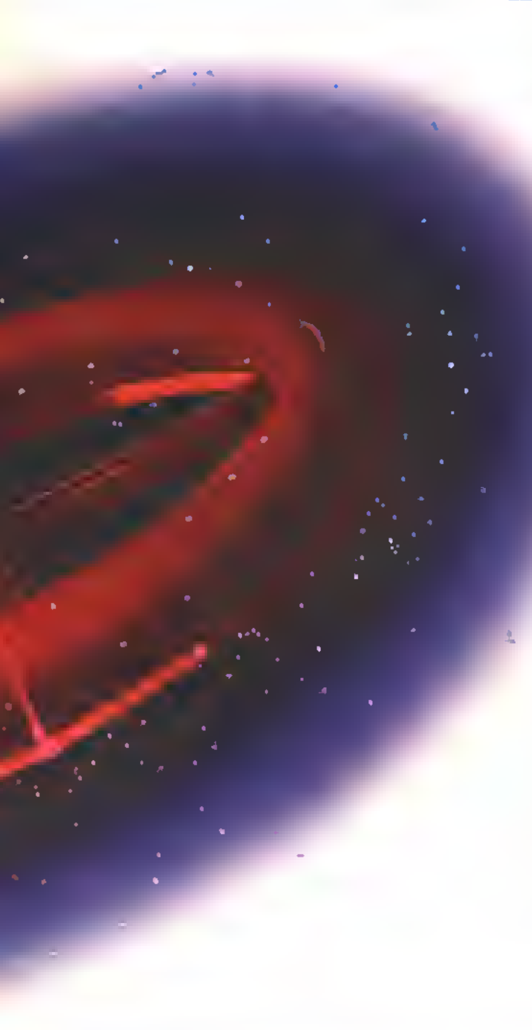
Entre las líneas 80 y 90 hemos incluido un bucle que produce el movimiento del *sprite*. Este bucle incrementa en uno las coordenadas xx e yy, tantas veces como indique la variable 1 y en la dirección que indiquen las variables DX y DY. Dentro del bucle, las líneas 82 y 83 definen unos límites para la pantalla, más allá de los cuales no puede moverse nuestro *sprite*.

Por último, al terminar el bucle, la línea 100 se encarga de volver al principio del mismo después de escoger nuevos valores aleatorios para L, DX y DY.

Como ves, mover tus *sprites* por la pantalla es algo muy sencillo. Sólo tienes que hacer uso de la instrucción PUTSPRITE.

ENEMIGOS MORTIFEROS Y EXTRATERRESTRES

- LAS RUTINAS PARA JUEGOS DE MARCIANITOS
- DIBUJO DE LOS ELEMENTOS
- INCORPORACION DE LOS ELEMENTOS



Desde Los Invasores, hasta los últimos juegos de marcianitos, los enemigos que atacan disparando siempre han sido un desafío. Aquí tienes la manera de crearlos e incorporarlos en una rutina compleja de juego.

Los juegos tendrán mejor aspecto si utilizamos algunas de las características de los gráficos de alta resolución de tu máquina, en lugar de servirte de los caracteres ordinarios. Los programas con gráficos de alta resolución serán más complicados que los que sólo emplean a los caracteres del teclado, pero ten por seguro que los resultados realmente merecen la pena.

Muchos juegos de marcianitos están basados en la presencia de enemigos invasores o extraterrestres que disparan contra tí, en vez de detenerse plácidamente a esperar que los aniquiles.

Seguidamente te presentamos un juego llamado Estación Espacial que te enseñará la manera de programar el movimiento aleatorio de un «invasor» por la pantalla, así como la forma de lanzar misiles contra un blanco.

El jugador dispone de 10 misiles con los que tendrá que destruir al marciano invasor. Este se mueve aleatoriamente por la pantalla descendiendo, desde la línea superior hacia la línea inferior, en la que se encuentra la nave del jugador.

Con los 10 misiles hay que destruir al marciano antes de que llegue a la línea de la nave. De no hacerlo, la nave del jugador quedará destruída y terminará el juego.

Tal como se presente aquí el juego, no está realmente completo, ya que le falta la puntuación y el cronometraje.

Pero esto se remedia fácilmente con los métodos que presentamos en los capítulos anteriores.

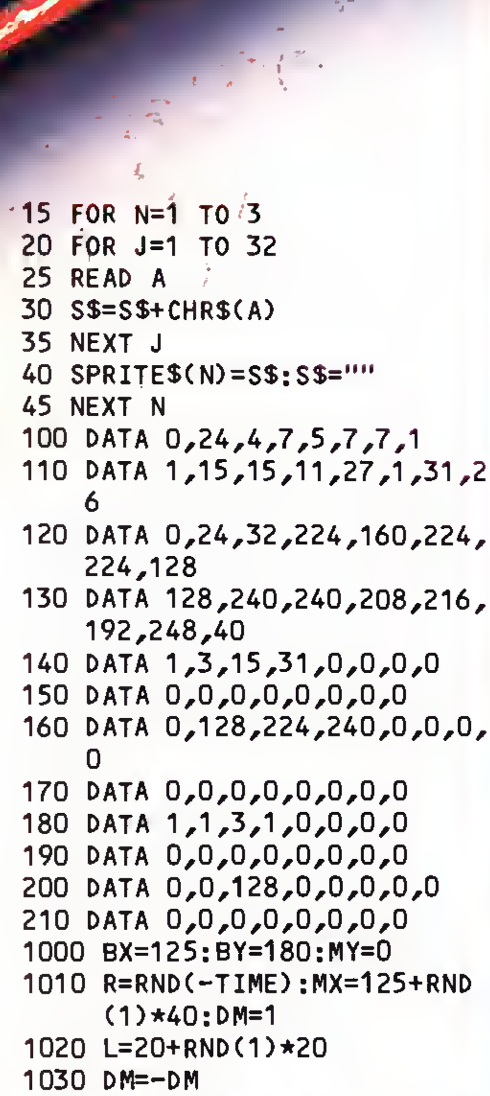
Por otro lado al estar escrito integramente en BASIC y sin cuidar demasiado de la velocidad, el juego puede resultar un poco lento. En cualquier caso es una primera versión en la que se ha pretendido aclarar conceptos sobre movimiento, disparo de misiles, etc.

Para conseguir mayor velocidad recurriremos, en próximos capítulos, a

la utilización de rutinas de código máquina.

La presente versión del juego de la estación espacial utiliza los *sprites*, cuya información está contenida en un gran número de sentencias DATA cerca del comienzo del programa.

```
10 SCREEN2,2:CLS:SPRITEON:
   NM=5
```



```
15 FOR N=1 TO 3
20 FOR J=1 TO 32
25 READ A
30 S$=S$+CHR$(A)
35 NEXT J
40 SPRITE$(N)=S$:S$=""
45 NEXT N
100 DATA 0,24,4,7,5,7,7,1
110 DATA 1,15,15,11,27,1,31,2
    6
120 DATA 0,24,32,224,160,224,
    224,128
130 DATA 128,240,240,208,216,
    192,248,40
140 DATA 1,3,15,31,0,0,0,0
150 DATA 0,0,0,0,0,0,0,0
160 DATA 0,128,224,240,0,0,0,
    0
170 DATA 0,0,0,0,0,0,0,0
180 DATA 1,1,3,1,0,0,0,0
190 DATA 0,0,0,0,0,0,0,0
200 DATA 0,0,128,0,0,0,0,0
210 DATA 0,0,0,0,0,0,0,0
1000 BX=125:BY=180:MY=0
1010 R=RND(-TIME):MX=125+RND
     (1)*40:DM=1
1020 L=20+RND(1)*20
1030 DM=-DM
```

```
1040 MY=MY+10:IF MY>163 THEN
     GOTO2000
1050 FOR J=1 TO L
1060 MX=MX+DM*2
1070 A=STICK(0)
1080 IF A=3 THEN BX=BX+3:IF
     BX>200 THEN BX=200
1090 IF A=7 THEN BX=BX-3:IF
     BX<0 THEN BX=0
1100 A$=INKEY$:IF A$=" " AND
     MI<>1 THEN MI=1:SX=BX:SY
     =180
1110 IF MI=1 THEN SY=SY-10:IF
     SY<0 THEN MI=0:NM=NM-1:
     SY=180:PUTSPRITE3,
     (-20,SY),15,3
1120 IF NM=0 THEN GOSUB 2060
1130 PUTSPRITE1,(MX,MY),15,1
1140 PUTSPRITE2,(BX,BY),15,2
1150 IF MI=1 THEN PUTSPRITE3,
     (SX,SY),15,3
1160 ONSPRITE GOSUB 2010
1170 NEXT J
1180 GOTO1020
2000 SCREEN0:CLS:PRINT"El
     marciano ha terminado
     contigo":END
2010 SCREEN0:CLS:PRINT "HAS
     MATADO AL MARCIANO":
     PRINT
2020 PRINT"OTRA PARTIDA (S/N)
     ?";
2030 B$=INKEY$:IF B$="" THEN
     2030
2040 IF B$="s" THEN RUN
2050 IF B$="n" THEN END ELSE
     GOTO 2030
2060 SCREEN0:CLS:PRINT"Se
     acabaron los misiles":
     END
```



La primera línea del programa nos situa en el modo de alta resolución, limpia la pantalla, activa la detección de colisiones entre *sprites* y fija el número de misiles en 5 (NM=5).

Entre las líneas 15 y 45 el programa lleva a cabo la definición de los *sprites*. Para ello, y mediante dos bucles, se lleva a cabo la lectura de las características de los *sprites*, cuya forma viene dada en las sentencias DATA comprendidas entre la línea 100 y la 210.

Cada cuatro líneas DATA corresponden a un *sprite*. En total hay tres: uno para el marciano, uno para la estación espacial de lanzamiento de misiles y el último que corresponde al misil.

En las líneas 1000 y 1010, se establecen las coordenadas iniciales de la estación (BX,BY) y del marciano (MX,MY). La coordenada X del marciano (MX) se elige aleatoriamente en la línea 1010.

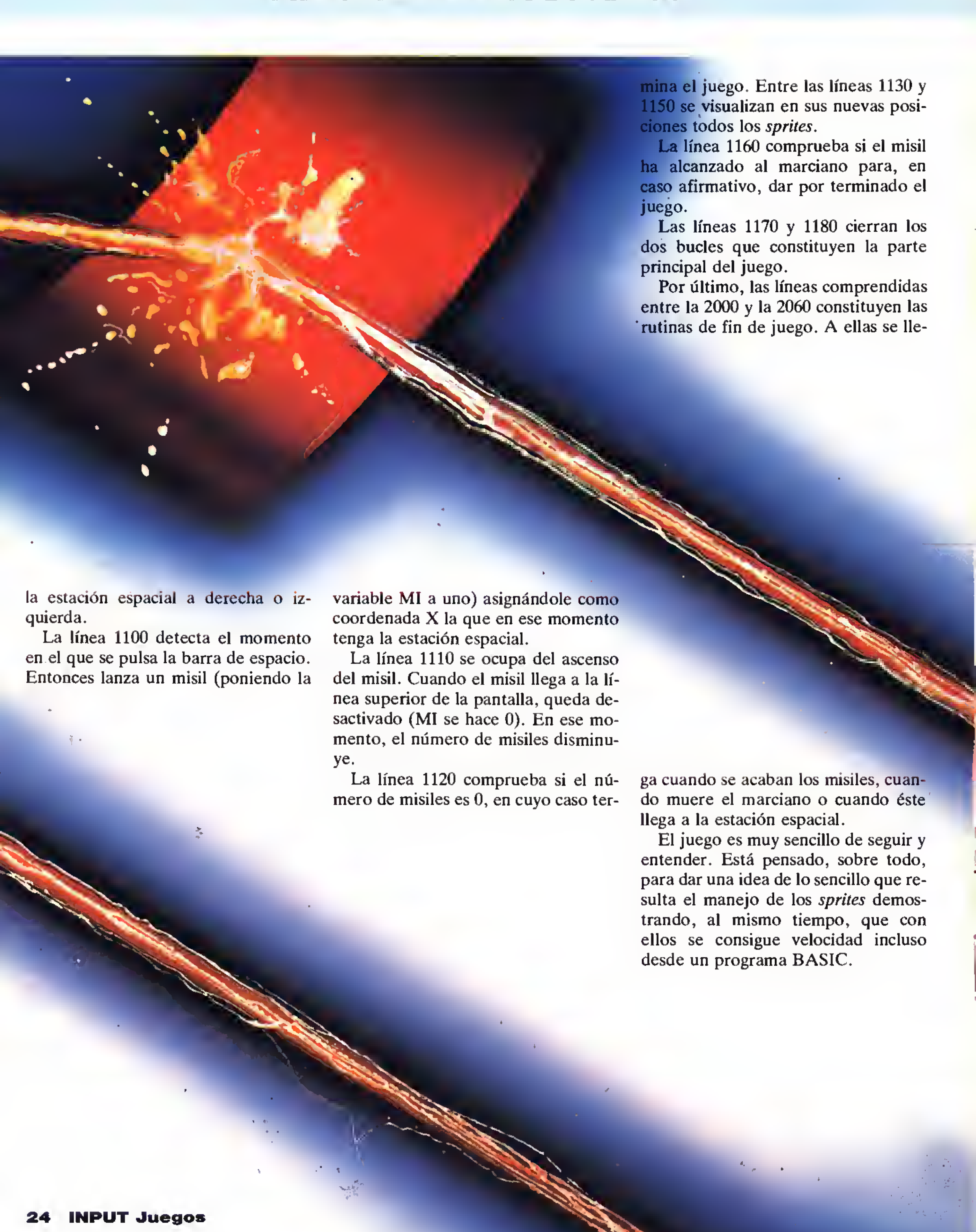
En las líneas 1020 y 1030 se elige aleatoriamente la longitud horizontal del desplazalmiento del marciano (L) y se alterna este desplazamiento de izquierda a derecha (al hacer DM=−DM). Es en la línea 1020 en la que se inicia el bucle principal del programa, dentro del que se mueven los *sprites*, se comprueba si ha habido colisiones, etc.



La línea 1040 hace descender al marciano al aumentar el valor de su coordenada Y (MY). Cuando esta vale más de 163, quiere decir que el marciano ha llegado a la estación espacial con lo que termina el juego.

Entre las líneas 1050 y 1070 hay otro bucle, que se encarga del desplazamiento horizontal del marciano al variar su coordenada X (MX) en la línea 1060.

Las líneas 1070 a 1090 se encargan de leer las teclas de cursor y desplazar



la estación espacial a derecha o izquierda.

La línea 1100 detecta el momento en el que se pulsa la barra de espacio. Entonces lanza un misil (poniendo la variable MI a uno) asignándole como coordenada X la que en ese momento tenga la estación espacial.

La línea 1110 se ocupa del ascenso del misil. Cuando el misil llega a la línea superior de la pantalla, queda desactivado (MI se hace 0). En ese momento, el número de misiles disminuye.

La línea 1120 comprueba si el número de misiles es 0, en cuyo caso termina el juego. Entre las líneas 1130 y 1150 se visualizan en sus nuevas posiciones todos los *sprites*.

La línea 1160 comprueba si el misil ha alcanzado al marciano para, en caso afirmativo, dar por terminado el juego.

Las líneas 1170 y 1180 cierran los dos bucles que constituyen la parte principal del juego.

Por último, las líneas comprendidas entre la 2000 y la 2060 constituyen las rutinas de fin de juego. A ellas se llega cuando se acaban los misiles, cuando muere el marciano o cuando éste llega a la estación espacial.

El juego es muy sencillo de seguir y entender. Está pensado, sobre todo, para dar una idea de lo sencillo que resulta el manejo de los *sprites* demostrando, al mismo tiempo, que con ellos se consigue velocidad incluso desde un programa BASIC.

CREACION DE NIVELES DE DIFICULTAD

- UN LABERINTO ALEATORIO
- OOS MANERAS OE CONVERTIR EL JUEGO EN MAS OIFICIL
- COMO MOVER AL JUGADOR
- AÑAOIENOO LA PUNTUACION

Algunos juegos de laberintos son muy fáciles de resolver, pero éste te dará dos niveles de dificultad y además presentará cada vez un laberinto diferente. Busca la manera de atravesarlo para encontrar el tesoro.

Los juegos de ordenador con frecuencia te piden que selecciones un nivel de dificultad antes de empezar a jugar. Esto hace posible que tanto principiantes como expertos puedan practicar el mismo juego, sin que sea demasiado difícil ni demasiado fácil para nadie.

Dependiendo de la naturaleza del juego, tienes muchas maneras de introducir el nivel de dificultad. Por ejemplo, puedes cambiar el número de enemigos, introducir una serie de retardos en el juego, permitir un tiempo mayor o menor, cambiar los obstáculos, etc.

En nuestro coleccionable ahora puedes ver cuántos niveles de dificultad se pueden introducir en un juego de laberintos. El juego utiliza una o dos formas de generar niveles de dificultad distintos. No se trata tan sólo de encontrar un camino para recorrer el laberinto, sino que además el jugador dispone de un tiempo fijo limitado en el que guiar a un hombre hasta algún tesoro que aparece dibujado en alguna parte del laberinto.

Para que la cosa resulte más difícil, puedes utilizar dos métodos. El primero consiste en cambiar la complejidad del laberinto. El otro es alterar el tiempo límite.

VIDAS

Cuando al jugador que está intentando alcanzar el tesoro, se le termine el tiempo, querrás imponerle algún tipo de penalización. Podrías hacer que el jugador perdiera algo de su puntuación, pero la penalización más ampliamente usada es hacer que pierda una vida.

Aquí el jugador recibe tres vidas, por lo que si no consigue encontrar el tesoro dentro del tiempo límite e tres ocasiones seguidas, el juego termina-

LABERINTOS ALEATORIOS

El juego de laberintos está basado en una subrutina de generación aleará.



toria de laberintos, que resulta un programa interesante por sí mismo, ya que dibuja cada vez un laberinto diferente, evitándote el tener que crear toda una serie de laberintos. En la página 14 vimos la forma de generar un laberinto con sentencias DATA y cómo incorporarlo en un programa; imagínate lo complicado que sería el tener que generar toda una serie de ellos.

El diseño de laberintos aleatorios es mucho más fácil que eso, pero más complicado de lo que te puedes imaginar. Una forma obvia de diseñarlos podría ser imprimir un número de bloques, por ejemplo gráficos incluidos en la ROM, aleatoriamente sobre la pantalla. Pero el problema es que podría resultar que no se obtuviera un laberinto, ya que no se garantiza que haya un camino a través del mismo; por ello, para usar este método habría que introducir alguna forma de comprobar que existe una salida.

COMO DIBUJAR LABERINTOS ALEATORIOS

La mejor manera de dibujar laberintos aleatorios es hacer un programa que dibuje una trayectoria aleatoria, y disponer la misma en forma de laberinto. El programa de tu máquina está diseñado de forma que la línea está contenida dentro de una trama dibujada sobre la pantalla. No se permite que la línea se cruce consigo misma en ningún caso. Cuando la trayectoria aleatoria ya no puede avanzar más —bien porque se encuentra con una esquina, o entre ella misma y la trama, o incluso puede quedar atrapada dentro de sí misma— el ordenador vuelve sobre sus pasos. Esto lo hace retrocediendo un paso cada vez y examinando la zona de alrededor. Cuando la máquina la encuentra, se inicia una nueva rama de la trayectoria aleatoria, por la cual continúa hasta que se tropieza de nuevo y empieza a retrasar sus pasos otra vez. El ordenador sigue intentando dibujar nuevas ramas hasta que la trama está llena, en cuyo caso vuelve al sitio donde empezó.

Después de que el programa ha terminado de dibujar el laberinto, sólo hay un camino posible para recorrerlo, que puede resultar muy fácil, ya que las ramas del camino no son complicadas. El laberinto también se puede recorrer con la «regla de la mano derecha» siguiendo la pared derecha (o la izquierda, según los casos) del laberinto durante todo el tiempo. Para evitar que alguien pueda hacer esto, te hacen falta «islas» en el laberinto, con las que interrumpir las paredes. Así, después de dibujar el laberinto, el programa dibuja una serie de bloques aleatorios que hacen que el laberinto parezca más complicado y que desconcertarán al que pretenda servirse de la regla de la mano derecha.

Cuando hayas introducido el programa completo, almacénalo (con SAVE) ya que en el siguiente artículo veremos la manera de añadirle algunos efectos sonoros.

Al ejecutar este programa (RUN) se te pide que elijas un nivel de dificultad. Puedes seleccionar un número entre 1, 2, 3 ó 4, que corresponden a duraciones de juego de 32, 24, 16 y 8 segundos. El objetivo del juego es alcanzar el tesoro, un asterisco situado al azar, en el menor tiempo posible. Tu «hombre» es un carácter # y para dirigirle hacia el blanco se utilizan las teclas de cursor.

```
39 'nivel de dificultad
40 SCREEN 0:CLS:WIDTH35
   :INPUT"Nivel de
   dificultad? (1-4) ";A
   :IF A<1 OR A>4
   THEN 40
50 SCREEN 1:CLS:WIDTH 30
   :TL=5-A:TL=TL*400:VI=3
   :DIM A(4)
69 'dibujo del laberinto
70 R=RND(-TIME):CLS:
   VPOKE BASE(6),&H44
   :KEY OFF
80 CO=BASE(5)
85 FOR J=0 TO 20:FI=CO+32*J
90 FOR N=FI+2 TO FI+30
   :VPOKE N,219:NEXT N
95 NEXT J
100 B=BASE(5)+49:A=B
    :VPOKE A,5
110 A(1)=-1:A(2)=-32
    :A(3)=1:A(4)=32
130 J=INT(RND(1)*4)+1:G=J
140 B=A+A(J)*2:IF VPEEK(B)
    =219 AND VPEEK(A+A(J))
    =219 THEN VPOKEB,J
    :VPOKE A+A(J),32:A=B
    :GOTO130
150 J=(J+1):IF J=5 THEN J=1
160 IF J<>G THEN GOTO 140
170 A=A-A(VPEEK(A))*2
    :IF VPEEK(A)=5 GOTO
    1000 ELSE GOTO130
999 'puntos aleatorios
1000 FOR Z=1 TO 10
1002 X=INT((RND(1)*19)+1)
     *32+INT(RND(1)*25)+4
1004 IF VPEEK(CO+X)=219
     THEN VPOKECO+X,32ELSE
     GOTO1002
1005 NEXT Z
1006 'colocacion tesoro
1007 X=(INT(RND(1)*20)+1)*
     32+(INT(RND(1)*27)+3)
     :IF VPEEK(CO+X)=219
     THEN 1007
1010 TE=CO+X:VPOKE TE,42
1012 TIME=0
1014 H=BASE(5)+49:HA=H
     :LOCATE 0,21:PRINT"PM"
     ;PM
1016 LOCATE 0,22:PRINT"VI"
     ;VI;" TI";:PRINT USING
     "#####";TIME/50;
     :PRINT" PU  ";PU
1018 IF TIME>TL THEN 2000
1020 'lectura teclado
1024 A=STICK(0)
1026 IF A=1 THEN H=HA-32
     :GOTO1034
1028 IF A=3 THEN H=HA+1
     :GOTO1034
1030 IF A=5 THEN H=HA+32
     :GOTO1034
1032 IF A=7 THEN H=HA-1
     :GOTO1034
1034 IF VPEEK(H)=42 THEN
     VPOKEH,32:VPOKE HA,32
     :GOTO 3000
1036 IF VPEEK(H)<>219 THEN
     VPOKE HA,32:VPOKEH,35
     :HA=H
1038 GOTO 1016
2000 VI=VI-1:FOR Z=1 TO
     100:VPOKE HA,Z:NEXT
```

```
      :BEEP:VPOKEHA,32
2002 IF VI>0 THEN 1012
2003 FOR J=1 TO 10:BEEP
      :NEXT:VPOKE TE,32:PU=0
2004 LOCATE0,22:PRINT
      "Otra vez (s/n)
      ":R$=INKEY$:IF R$=""
      THEN 2004
2006 IF R$="s" THEN 2012
2008 IF R$="n" THEN SCREEN 0
      :LOCATE 0,0:PRINT
      "Programa terminado"
      :END
2010 GOTO 2004
2012 LOCATE0,22:PRINT
      "Otro laberinto? (s/n)
      ":R$=INKEY$:IF R$=""
      THEN 2012
2014 IF R$="s" THEN VI=3
      :GOTO 70
2016 IF R$="n" THEN VI=3
      :GOTO 1006
2018 GOTO2012
3000 PU=INT(PU+100-TIME/50)
      :IF PU>PM THEN PM=PU
3010 GOTO 1006
```

El programa comienza por pedirnos el nivel de dificultad del juego. Dicho nivel, que se introduce en la línea 40, determina el valor del tiempo límite de cada vida, mediante la variable TL de la línea 50. La relación es inversa, es decir, cuanto más pequeño es el nivel de dificultad, mayor es el tiempo límite disponible. Además, en estas líneas se asigna el modo de pantalla SCREEN 1, se fija en tres el número de vidas (variable VI) y se dimensiona la matriz A que utilizaremos más adelante.

La verdadera rutina de creación del laberinto ocupa las líneas que van de la 69 a la 170. Vamos a comentar a fondo esta rutina, ya que hace uso de algunas interesantes posibilidades del ordenador, al trabajar directamente con la VRAM. La rutina comienza en la línea 70 limpiando la pantalla, eliminando la línea de teclas de función e introduciendo el valor &H44 en el primer byte de la tabla 6. Esta tabla controla los colores de los caracteres en el modo SCREEN 1. Al introducir el valor &H44 en el primer byte de la tabla 6, conseguimos que los caracteres correspondientes a los códigos 0 a 7, aparezcan en la pantalla en color azul sobre fondo azul, es decir, no se van a ver, pero aunque no se vean van a estar ahí y nos van a ayudar a construir el laberinto. A continuación, entre las líneas 80 y 95, se dibuja el fondo del laberinto, un rectángulo de 21 filas × 28 columnas. Este rectángulo se dibuja POKEando el valor 219 (que corresponde a un cuadrado) en la zona de VRAM correspondiente a la pantalla, es decir, a partir de la dirección de comienzo de la tabla 5 (BASE(5)). Si quieres experimentar, puedes cambiar los límites de los bucles FOR...NEXT para obtener un laberinto más grande o más pequeño, puedes también cambiar el valor 219 por otro, lo que hará que cambien los caracteres que definen el fondo y, por último, puedes cambiar de color actuando sobre el byte correspondiente de la tabla 6. Recuerda para ello que cada uno de los 32 bytes de la tabla 6 determina el color de 8 caracteres (el byte cero de los caracteres de códigos 0 a 7, el 1 de los de códigos 8 a 15, etc).

Una vez dibujado el fondo, se elige la posición de inicio del laberinto (en la línea 100). Esta posición corresponde a la dirección BASE(5) + 49 de la VRAM. Si quieres cambiarla no tienes más que escoger un valor distinto de 49. En dicha posición se POKEa el caracter de código 5, pero como hemos dicho que aparece en azul y sobre fondo azul, no podremos verlo. En la línea 110 se definen los 4 posibles movimientos de la traza del laberinto. A(1) y A(3) corresponden a movimientos a izquierda y derecha respectivamente, mientras que A(2) y A(4) representan las direcciones arriba y abajo.

Las líneas comprendidas entre la 130 y la 170 dibujan el laberinto aleatorio de la siguiente forma: primero, en la línea 130, se elige una dirección

aleatoria entre 1 y 4 (arriba,abajo, izquierda o derecha). En dicha dirección vamos a intentar dibujar el laberinto. Para ello comprobamos, en la línea 140, si podemos avanzar 2 caracteres en la dirección elegida. Si en las dos posiciones, en la dirección considerada, nos encontramos con el código 219, quiere decir que estamos sobre el fondo del laberinto y que podemos dibujar la traza. En este caso imprimimos un espacio (código 32) en el primer caracter, mientras que en el segundo, y aquí está el truco, imprimimos la dirección de movimiento (1,2,3 o 4). Como esta dirección se imprime en azul sobre fondo azul, no se ve, pero está ahí y nos va a permitir volver sobre nuestros pasos, retrocediendo a través del laberinto. En el caso de que no podamos avanzar en la dirección elegida, al salirnos de los límites del laberinto o bien porque haríamos que la traza se cruzara a sí misma, se elige la siguiente dirección (en la línea 150) y se intenta de nuevo (a través del salto a la 140, desde la línea 160).

Cuando se hayan comprobado las 4 posibles direcciones y no haya sido posible avanzar, pasaremos a la línea 170. Ella se encarga de hacernos retroceder. Para ello y recordando que habíamos dejado escrita, en cada punto, la dirección por la que veníamos, no hay más que moverse en la dirección contraria. De esta forma se retroceden 2 posiciones en el laberinto y se salta de nuevo a la línea 130, para continuar con el laberinto en otra dirección. Cuando el laberinto esté terminado, iremos retrocediendo, posición a posición, hasta llegar a la posición de partida. La línea 170 detectará la llegada a la posición inicial ya que el VPEEK de esta posición es 5 (lo habíamos escrito para este fin en la línea 100). Así pues, la linea 170 detectará que se ha terminado de dibujar el laberinto y dará paso a la siguiente parte del programa que comienza en la línea 1000. Esta y las siguientes, hasta la 1005, se ocupan de colocar aleatoriamente 10 espacios sobre el laberinto, estableciendo así 10 pasos o túneles a través de las paredes del mismo. Si quieres que aparezcan más o menos, no tienes más que cambiar el valor 10, en la línea 1000, por otro valor. Incluso puedes hacer que el número dependa del nivel de dificultad elegido, por ejemplo, a mayor nivel de dificultad, más túneles. Para ello tendrás que cambiar el valor 10 de la línea 1000 por una variable cualquiera que tomará su valor en función del nivel de dificultad elegido. Siguiendo con el programa, las líneas que van de la 1006 a la 1010, se ocupan de elegir una posición aleatoria en la que que colocar el tesoro (representado por un asterisco, de código 42).



A partir de este momento comienza la cuenta atrás. El programa, en las líneas 1012 a 1016, pone a cero el reloj interno y se ocupa de imprimir, en las 2 líneas inferiores de la pantalla, los valores de puntuación máxima (PM), vidas (VI), tiempo transcurrido (TIME) y puntuación (PU).

Mientras no se alcance el tiempo límite, lo que se comprueba en la línea 1018, el programa trabajará en la rutina de lectura de teclado y actualización de la posición del husca tesoros. Esta ocupa las líneas comprendidas entre la 1020 y la 1038. Cada vez que el buscador alcance el tesoro, la línea 1034 cederá control a la línea 3000. Esta y la 3010 se encargan de actualizar la puntuación, devolviendo el control a la línea 1006 para que aparezca un nuevo tesoro.

En caso de cumplirse el tiempo límite, el programa se dirige a la línea 2000. En esta línea se inicia la rutina de comprobación de vidas y de fin de juego. Mientras VI sea mayor que cero (es decir mientras haya vidas) el programa seguirá ofreciendo tesoros. Cuando se acaben las vidas entrará en acción la rutina que empieza en la línea 2003 encargada de hacernos, escoger entre dejar de jugar, jugar otra vez en el mismo laberinto o jugar otra vez pero en un laberinto nuevo.

ROMPIENDO LA BARRERA DEL SONIDO

- CREANDO EL SONIDO CORRECTO
- EFECTOS ESPECIALES PARA EL LABERINTO
- RUIDOS DE EXPLOSIONES, SILBIDOS Y DISPAROS

Puedes darles vida a tus juegos de ordenador y hacerlos más excitantes añadiéndoles algunos efectos sonoros; vale todo, desde pitidos, explosiones y el ruido de los disparos de los invasores, hasta una marcha fúnebre o un tren de vapor.

Normalmente los programas de juegos suelen llevar incorporados toda clase de efectos sonoros que los hacen más excitantes: explosiones, disparos, pequeñas melodías o cualquier otra cosa que permita la imaginación del programador.

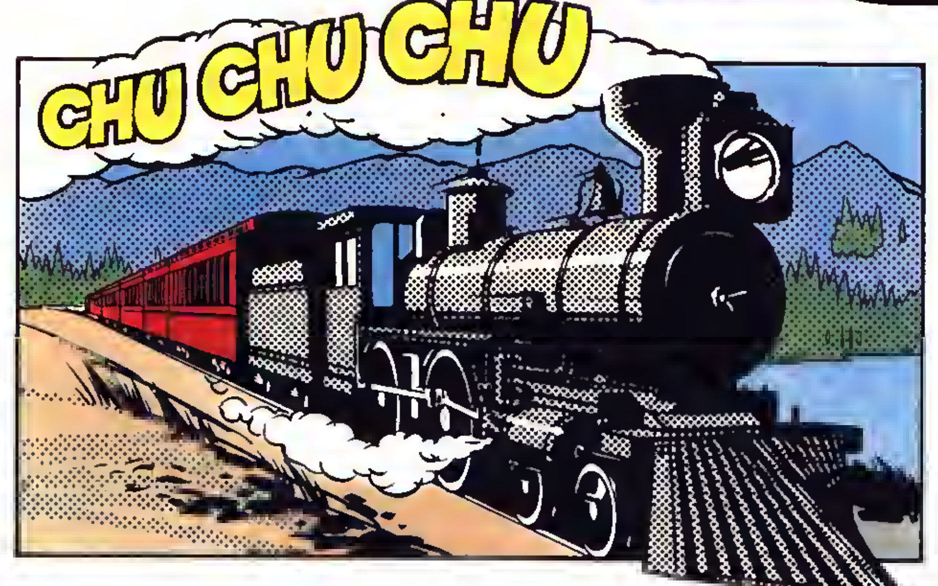
En este capítulo de nuestro coleccionable te presentamos un pequeño repertorio de efectos sonoros prefabricados, que puedes utilizar tal como vienen aquí o como base para nuevos experimentos.

Recuerda que no existen reglas infalibles y recetas rápidas para producción de efectos sonoros. Si lo que tu juego necesita es un sonido para acompañar una escena en la que se ve a un sujeto al que están zurrando, no tendrás más remedio que sentarte ante tu máquina y hacer diversas pruebas. Recíprocamente, un sonido aparentemente sin sentido puede resultar magnífico si le encuentras el conjunto de gráficos adecuados.

La forma de incorporar los efectos sonoros en tus programas depende de su complejidad y de la frecuencia con que los vayas a utilizar. Para efectos sencillos que sólo se utilizan una vez en el programa, puede ser suficiente una estructura del tipo IF ... THEN, pero para los más complejos o que se utilizan varias veces, es mejor escribir una subrutina. La sofisticación de los efectos que puedas conseguir, depende parcialmente de tu propia habilidad de programación, pero también está determinada por las capacidades sonoras de tu ordenador. Por ejemplo tu **MSX** tiene un generador de sonidos, capaz de sintetizar una enorme cantidad de efectos diferentes.

El generador de sonidos es extremadamente sofisticado y dispone de tres «voces» o canales de sonido fácilmente controlables. Se puede conseguir una gran variedad de sonidos,





empezando con simples efectos sonoros que pueden incorporarse dentro de cualquier programa de juegos.

Se pueden utilizar una, dos o tres voces para conseguir estos efectos y enseguida veremos la forma en que pueden ser combinados. En un próximo capítulo de la sección de programación BASIC, describiremos detalladamente la forma de programar eficazmente el *chip* **AY-3-8910**.

Este *chip* es lo bastante potente como para permitir la creación de una gran variedad de efectos sonoros interesantes con una cantidad de programación relativamente pequeña. En lo que sigue vamos a centrarnos en el

manejo directo del *chip* a través de sus registros y del uso de la instrucción SOUND. Esta instrucción, cuya sintaxis es:

```
SOUND n.registro,valor
```

sirve para colocar determinados valores en los distintos registros del *chip*. Su utilización es a veces necesaria pues permite llegar a donde no llega la instrucción PLAY. Por otro lado resulta muy interesante ver las grandes diferencias de sonido que se pueden conseguir alterando, aunque sólo sea ligeramente, los contenidos de algunos de los registros. Te recomendamos que para trabajar con los registros tengas delante una tabla de los mismos, como la que publicamos en la página 11 del número uno de **INPUT**.

Para cada sonido se requieren en general hasta cinco o seis asignaciones a determinados registros. Para valorar los efectos de cada una teclea el siguiente programa:

```
100 SOUND6,30
105 SOUND8,16
110 SOUND7,&B10110111
112 SOUND12,45:SOUND11,0
115 SOUND 13,14
120 FOR J=1 TO 5000:NEXT
130 GOTO 100
```

Ejecuta el programa (RUN) y podrás oir el inconfundible rumor de las olas rompiendo contra el acantilado. Vamos a ver cómo actúa cada registro. La primera asignación, al registro 6, determina el periodo de la señal de ruido, que es la que da al sonido que escuchas ese característico timbre de «ola de mar». El valor asignado a este registro (que puede ser un número de 0 a 63) es en este caso 30. Podemos decir que es responsable del tamaño de las olas. Prueba a introducir un valor más pequeño y verás como parece que rompen olas más pequeñas. Y lo mismo, prueba a introducir un valor más grande y escucharás el rumor de olas de tremendas dimensiones.

El siguiente registro, el 8, controla la mezcla de la señal del canal A con el generador de envolventes. La siguiente asignación es el registro 7. Este controla la asignación de ruido o tono a cada uno de los 3 canales de sonido A, B y C. En este caso, y sin considerar los 2 bits de la izquierda que tienen otros cometidos, el 0 del cuarto bit determina que por el canal A se escuche la señal de ruido.

Los registros 11 y 12 determinan el periodo de repetición de la envolvente. En este caso, esta repetición determina algo así como la cadencia de las olas. Introduce en 12 un valor mayor y verás cómo la frecuencia de llegada de olas disminuye. Por último el registro 13 fija la forma de la envolvente. En nuestro ejemplo hemos escogido una envolvente triangular.



Vamos a ver cómo varía el sonido con unas ligeras modificaciones en los registros. Teclea lo que sigue:

```
100 SOUND6,30
105 SOUND8,16
110 SOUND7,&B10110111
112 SOUND12,4:SOUND11,0
115 SOUND 13,14
120 FOR J=1 TO100:NEXT
130 GOTO 100
```



Escribe RUN y escucha. Las olas han dado paso a un tren a vapor. Fíjate que hemos utilizado los mismos registros, y que en algunos de ellos no hemos cambiado el valor. Lo que sí tienes que tener en cuenta es que, además de los registros estamos utilizando el bucle de retardo de la línea 120. Prueba a hacer experimentos. Cambia el valor del registro 12 y verás cómo cambia la velocidad del tren.

SONIDOS PARA EL LABERINTO

Si tienes almacenado el juego de laberintos que vimos en el capítulo anterior, puedes añadirle ahora las siguientes líneas después de cargar (LOAD) el programa.

```
10 SOUND 1,0
12 SOUND 8,0
14 SOUND 7,&B11111110
2000 VI=VI-1:SOUND 8,15
     :FOR Z=1 TO 100:VPOKE
     HA,Z:SOUND 0,2*Z:FOR
     ZZ=1 TO 10:NEXT:NEXT
     :VPOKE HA,32:SOUND8,0
3005 SOUND 8,16:SOUND 13,1
     :SOUND 0,60:SOUND 1,0
     :SOUND 11,230:SOUND
     12,35:
```

Las tres primeras líneas se encargan de inicializar el *chip* de sonido al comienzo del juego. La nueva línea 2000, que sustituye a la línea 2000 del programa del capítulo anterior, incluye la generación de un tono, por el canal A, cuyo periodo va aumentando (el sonido se va haciendo más grave) a medida que se POKEan valores en el registro 0. Por su parte, la línea 3005, crea una especie de ¡PING! cada vez que el buscador de tesoros llega a su objetivo antes de que se cumpla el tiempo límite.

Prueba estos sonidos y di lo que te parecen. Verás como, aún tratándose de sonidos muy sencillos y fáciles de programar, le dan mucha gracia al juego. He aquí otros efectos de sonido con los que te gustará experimentar y que podrás incorporar a cualquiera de tus juegos. El primero es el sondio de un helicóptero.

```
5 I=2.5
10 SOUND 7,&B00110111
20 SOUND 8,14
30 FOR L=63 TO 1 STEP -I
   :SOUND 6,L:NEXT L
```

```
35 A$=INKEY$:IF A$="q"
   THEN I=I+.1:IF I>6.5
   THEN I=6.5
36 IF A$="z" THEN I=I-.1
   :IF I<1.5 THENI=1.5
40 GOTO 10
```

El sonido del helicóptero se consigue mediante el generador de ruido, que ha sido asignado al canal A. Desde el bucle de la línea 30 se hace variar rápidamente el periodo de la señal de ruido (actuando sobre el registro 6) lo que produce un sonido muy similar al de un helicóptero. Las líneas 35 y 36 te permitirán, actuando sobre las teclas **q** y **z**, acelerar o decelerar el helicóptero. Esto te puede dar una idea de cómo modificar los parámetros de un sonido desde el interior de un programa. ¿Imaginas lo bien que quedaría este sonido si sobre la pantalla apareciera la imagen un helicóptero? Podrías hacer que al mismo tiempo que el sonido se acelera, tu helicóptero iniciara el ascenso hacia la parte superior de la pantalla.

Vamos con otro ejemplo.

```
10 N=240
20 SOUND 7,&B00111110
30 SOUND 8,14
40 SOUND 0,170:SOUND 1,0
50 FOR J=1 TO N:NEXT
60 SOUND 0,150:SOUND 1,0
70 FOR J=1 TO N:NEXT
80 A$=INKEY$:IF A$="q"
   THEN N=N+30
90 IF A$="z" THEN N=N-30
   :IF N<1 THEN N=1
100 GOTO 40
```

Al escribir RUN podrás escuchar el sonido de la sirena de un coche de la policía. La rutina utiliza en este caso uno de los generadores de tono, el del canal A. El generador de ruido no interviene. En la línea 40 se fija un periodo para el tono, el que corresponde al ciclo grave de la sirena. Durante el tiempo que dura el bucle de la línea 50 suena el tono grave. A continuación, en la línea 60, se fija un nuevo periodo, esta vez el del ciclo agudo. Al alternar estos dos ciclos, grave y agudo, se produce una buena imitación del sonido de la sirena. Por último, las líneas 80 y 90 se ocupan de modificar la duración de cada uno de los ciclos cuando se pulsan las teclas **q** y **z**. Prueba al pulsarlas y verás como cambia la duración de cada ciclo de la sirena.

El ejemplo que te vamos a presentar ahora, está indicado para los juegos en los que se utiliza un arma laser. Teclea lo que sigue:

```
5 FOR J=150 TO120 STEP -3
10 SOUND0,J:SOUND 1,1
20 SOUND6,10
30 SOUND7,&B00110110
40 SOUND 8,16
50 SOUND 11,100:SOUND 12,8
60 SOUND 13,4
70 FOR K=1 TO 170:NEXT
80 NEXT
```

En este caso se escucha una mezcla, a través del canal A, de las señales de uno de los generadores de tono y del generador de ruido blanco. La mezcla de estas dos señales se modela en volumen mediante una envolvente en rampa (esta envolvente es la número 4 de las predefinidas en el *chip*). La envolvente es la responsable de que la ráfaga de la laser vaya aumentando paulatinamente de volumen, hasta llegar a su volumen máximo, momento en el que el sonido se corta bruscamente. Por último, el bucle FOR...NEXT, de las líneas 5 y 80, determina que el periodo del tono vaya disminuyendo a cada ráfaga. De esta forma el sonido del laser resulta mucho más sideral e intergaláctico.

Por último, te ofrecemos una rutina de sonido en la que se utilizan valores aleatorios para el periodo de las señales de los tres generadores de tono.

```
10 SOUND 7,&B00111000
20 SOUND 8,15:SOUND 9,15
   :SOUND 10,15
30 R=RND(-TIME)
40 F1=10+(90*RND(1))
   :F2=50+(80*RND(1))
   :F3=180+(50*RND(1))
50 SOUND 0,F1:SOUND 2,F2
   :SOUND 4,F3
60 FOR J=1 TO 40:NEXT
70 GOTO 40
```





PROGRAMANDO AVENTURAS

- QUE ES UN JUEGO DE AVENTURAS
- COMO JUGARLO
- SUGERENCIAS PARA
- RESOLVER AVENTURAS
- ESCRIBE TU PROPIO JUEGO

Transporta a tus amigos a un mundo de fantasía de tu propia creación, y proporciónales algún quebradero de cabeza. Nos asomamos al mundo y a la historia de los juegos de aventuras.

Para los que queráis descansar un poco de los juegos de marcianitos con disparos, exite una alternativa: los juegos de aventuras. En este tipo de juego el participante se ve totalmente inmerso en un mundo de fantasía creado por el programador. Ejercitando su buen juicio, su inteligencia y su conocimiento de los hechos y personajes raros que encuentre, viaja por un mundo de fantasía intentando completar la búsqueda imaginada por el programador.

En los próximos números de **INPUT MSX** aprenderás la manera de escribir tus propios juegos de aventuras, pero primero veremos una introducción a estos juegos y en qué consisten.

LA HISTORIA DE LAS AVENTURAS

La idea de escribir juegos de aventuras procedía originalmente de la popularidad de juegos de ordenador tales como **Dragones y Mazmorras**, y el deseo de utilizar los ordenadores para algo más que el mero proceso de datos.

En **Dragones y Mazmorras**, el jugador adopta una determinada personalidad y penetra (con su imaginación) en un mundo conocido como la **Mazmorra**, creado por el **Carcelero**. En los juegos de aventuras el programador adopta un papel similar al del **Carcelero**, creando un mundo propio. Por su parte el jugador juega un papel semejante al de alguno de los personajes del juego.

A diferencia del juego tradicional, los jugadores de aventuras por ordenador no pueden elegir normalmente los rasgos de su carácter. De ello se ocupó el programador a la hora de escribir la aventura. En algunas de las versiones más sofisticadas, pueden seleccionar realmente su equipo, etc., antes de empezar su búsqueda. Quizá resulten algo menos sedientos de sangre, aunque naturalmente esto se deja al criterio del autor.

La primera aventura se escribió en un gran ordenador, y no se utilizó el BASIC sino el FOR-



TRAN. El programa ocupaba 300K de memoria, algo más que lo que lleva incorporado tu microordenador.

Sin embargo, el verdadero principio con los micros fué debido a **Scott Adams**, que trasladó algunas ideas al célebre **TRS 80** en 1978, demostrando que era totalmente factible escribir un juego de aventuras satisfactorio precisando menos espacio de memoria. Desde entonces los temas de aventuras que **Adams** adaptó para sus juegos —**Aventurlandia**, **La Cueva del Pirata**, **El Misterio de la Casa Encantada**— han sido utilizados múltiples veces.

TIPOS DE AVENTURAS

Tanto el juego original para un gran ordenador, como los juegos de **Adams** para microordenador únicamente presentan texto sobre la pantalla. Este tipo de aventuras, en las que no aparece ningún tipo de gráfico, sólo textos, siguen siendo las más populares, y hay quien afirma que son los mejores tipos de aventura.

Las aventuras de texto existen realmente en la mente del jugador.Cuando juegues con una de estas buenas aventuras de texto, te verás totalmente envuelto en la historia.

Los gráficos tienen que ser muy sofisticados para poder competir con tu imaginación. Por ejemplo, es posible que te imagines un ogro mucho más feroz que cualquiera que pueda salir incluso de la mejor de las pantallas gráficas, por lo que es muy posible que los gráficos echen a perder tu disfrute. Otra importante consideración en contra de los gráficos es que la pantalla requiere una gran cantidad de memoria, que en otro caso podría servir para alargar más la aventura. Además es posible que resulte demasiado lenta, debido a que el jugador tiene que esperar a que se dibuje una nueva imagen cada vez que se cambia de situación.

Algunas aventuras dan cierta puntuación al completar determinadas etapas, de forma que si te matan en algunas puedas juzgar lo bien o mal que lo has hecho. Algunas incluso te dan una categoría, por ejemplo novato o experto. En el otro extremo de la escala están las aventuras en que no se te da ninguna clave sobre si lo estás haciendo bien o mal, o cuánto te falta hasta la meta final. La última satisfación procede del hecho de resolver una serie de rompecabezas sin fin y de ir acercándose cada vez más hasta encontrar el final del asunto y resolver la aventura.

JUGANDO A LAS AVENTURAS

Cuando ejecutas una aventura, normalmente el programa te dice algo acerca del mundo en el que te vas a encontrar, puede ser en algún paraje exótico de la Tierra, un planeta de una galaxia lejana o en un mundo sólo existente en la fantasía. El juego puede desarrollarse en el pasado, el presente o el futuro, o incluso en una mezcla de los tres. Normalmente te dará unas cuantas indicaciones informativas de base que te servirán de ayuda, tales como quién manda en ese mundo, quién eres tú (si has asumido una determinada personalidad), algo sobre tus amigos y enemigos, y lo que es más importante, lo que tienes que hacer para resolver la aventura y ganar el juego. Lee con cuidado las instrucciones, ya que normalmente contienen mucha información importante.

Después de todo esto, aparecerá la primera descripción del lugar. Probablemente te dirá algo como esto:

TE ENCUENTRAS CERCA
DE UNA ENORME
OLLA LLENA HASTA LOS
BORDES DE UN
ESPUMEANTE
LIQUIDO VERDE.
HAY UN OLOR MALIGNO
EN EL AMBIENTE.
EN EL SUELO
HAY UNA GRAN
CUCHARA.
PUEDES IR HACIA
EL ESTE,
OESTE, NORTE

¿QUE HACES AHORA?

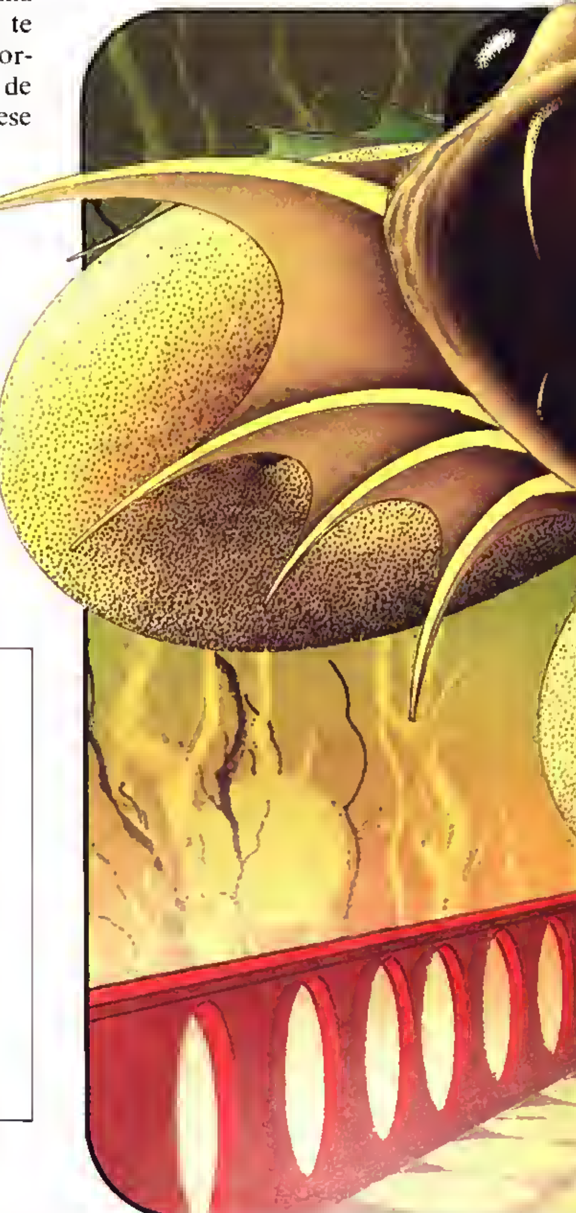
Tienes que decidir lo que quieres hacer. ¿Usarás la cuchara para remover el líquido, o incluso para intentar beber algo? ¿La dejarás ahí? ¿O te dedicarás a explorar, buscando una botella o algún otro recipiente para poder llevarte un poco de líquido verde?

Si te decides a usar la cuchara, tendrás que teclear algo así:

COGER CUCHARA

a lo que el ordenador replicará OK (muy bien), o tal vez NO PUEDES COGER CUCHARA, TODAVIA!, o cualquier otro mensaje.

En cada etapa del juego tienes que decirle al ordenador exactamente lo que quieres hacer, cómo se lo dices



depende del juego. Casi todos los juegos esperarán que comuniques tus instrucciones al ordenador con un verbo en infinitivo seguido de un nombre, por ejemplo, COGER CUCHARA, ESTRANGULAR ELEFANTE, ARRANCAR ARBOL, etc.

Otros juegos más sofisticados aceptarán frases completas, pero esto es más bien la excepción que la regla. Dicha clase de juegos permite decir algo como MATA A ESE PESADO INSECTO DANDOLE UN PISOTON MIENTRAS CANTAS «ALL YOU NEED IS LOVE» (popular canción de los **Beatles**). Un programa que acepte instrucciones tan complicadas como ésta, tendrá que ser forzosamente muy complejo y está fuera de los objetivos de un principiante.

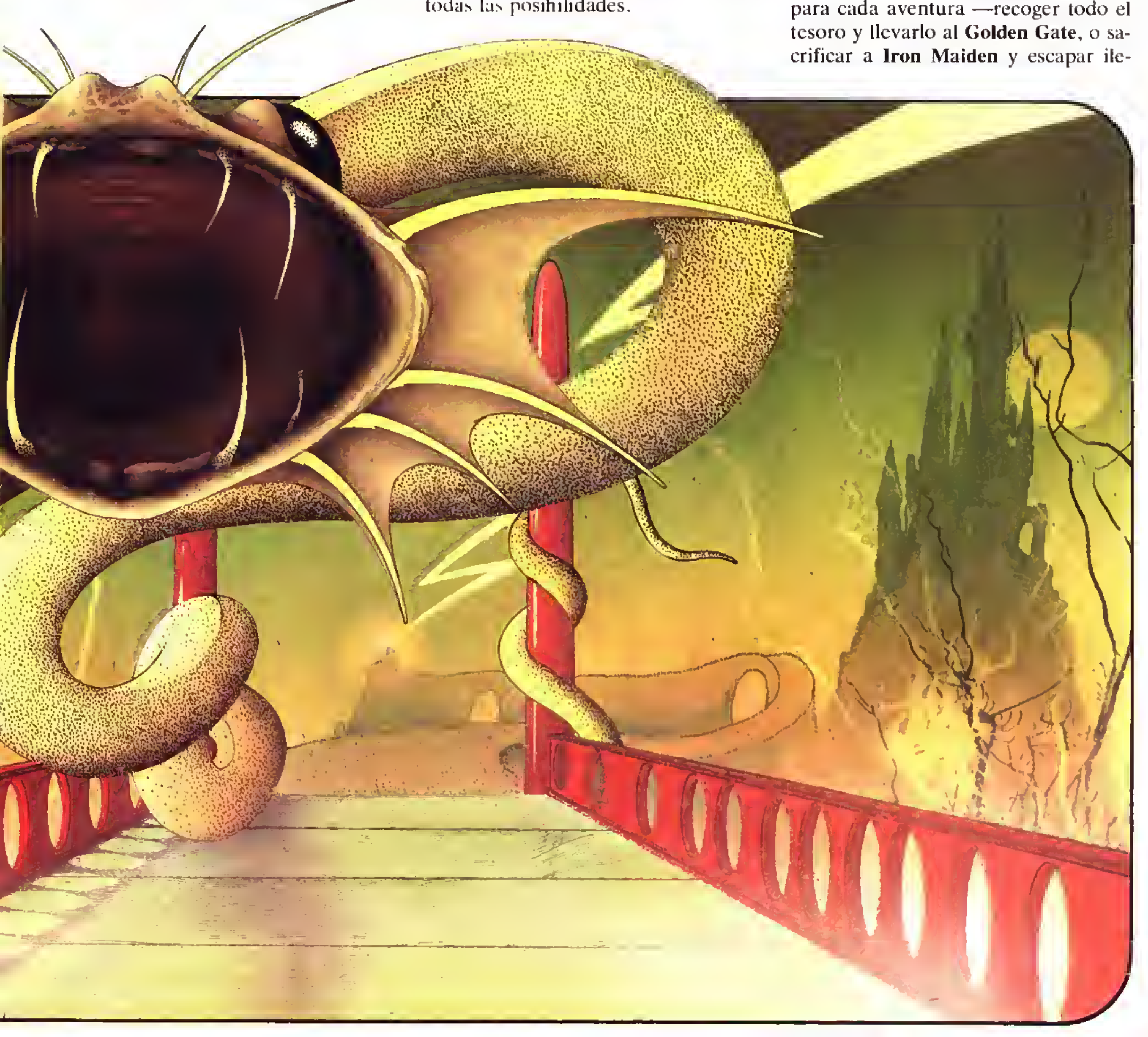
La mayoría de los juegos de aventuras entenderán e incluso esperarán versiones abreviadas de las palabras. Por ejemplo, en los juegos de aventuras es muy normal teclear N en lugar de Norte. Utilizando este tipo de abreviaturas puedes agilizar el juego y ahorrar espacio de memoria.

Las direcciones pueden no ser precisamente N, S, E y O. Podrias encontrarte con ARRIBA o ABAJO, o incluso con NE, SE, SO y NO. Si el juego no te dice qué direcciones tienes disponibles, no te olvides de probar todas las posihilidades.

El caso más normal es que el mundo de la aventura esté basado en una retícula de posibles lugares, habitualmente un cuadrado. Lo que puedan representar estos lugares queda a capricho del programador, pueden ser las estancias de un castillo, o las cámaras subterráneas de una mina. El eslabón de unión entre distintos lugares puede ser algo ohvio, como una puerta o un tramo de escaleras, o puede estar menos claro, por ejemplo un río que tienes que atravesar a nado.

RESOLVIENDO AVENTURAS

Normalmente sólo hay una solucion para cada aventura —recoger todo el tesoro y llevarlo al **Golden Gate**, o sacrificar a **Iron Maiden** y escapar ile-



so— y una secuencia fija de problemas que resolver. Lo más probable es que necesites muchos, muchos intentos para resolver el juego antes de que termine la aventura. De hecho toda aventura que no te requiera días, semanas o incluso meses de sudores, no es muy buena.

Existen algunas reglas y sugerencias básicas que te ayudarán a resolver un poco más rápidamente la mayoría de las aventuras.

Casi sin excepción, todos los objetos que encuentres en las aventuras tienen algún uso. Para el programador supondría un gasto de memoria innecesario llenar todo de objetos que luego nadie va a utilizar, pero ten cuidado: algunos objetos podrían ser «armas de doble filo». Por ejemplo, es posible que necesites llevar una bolsa con monedas de oro para pasar un puente de peaje, pero si te decides a pasar el río a nado, su peso podría hacer que te hundieras. En general, coge todos los objetos que puedas, aunque a veces te encontrarás que sólo puedes acarrear un número determinado de objetos.

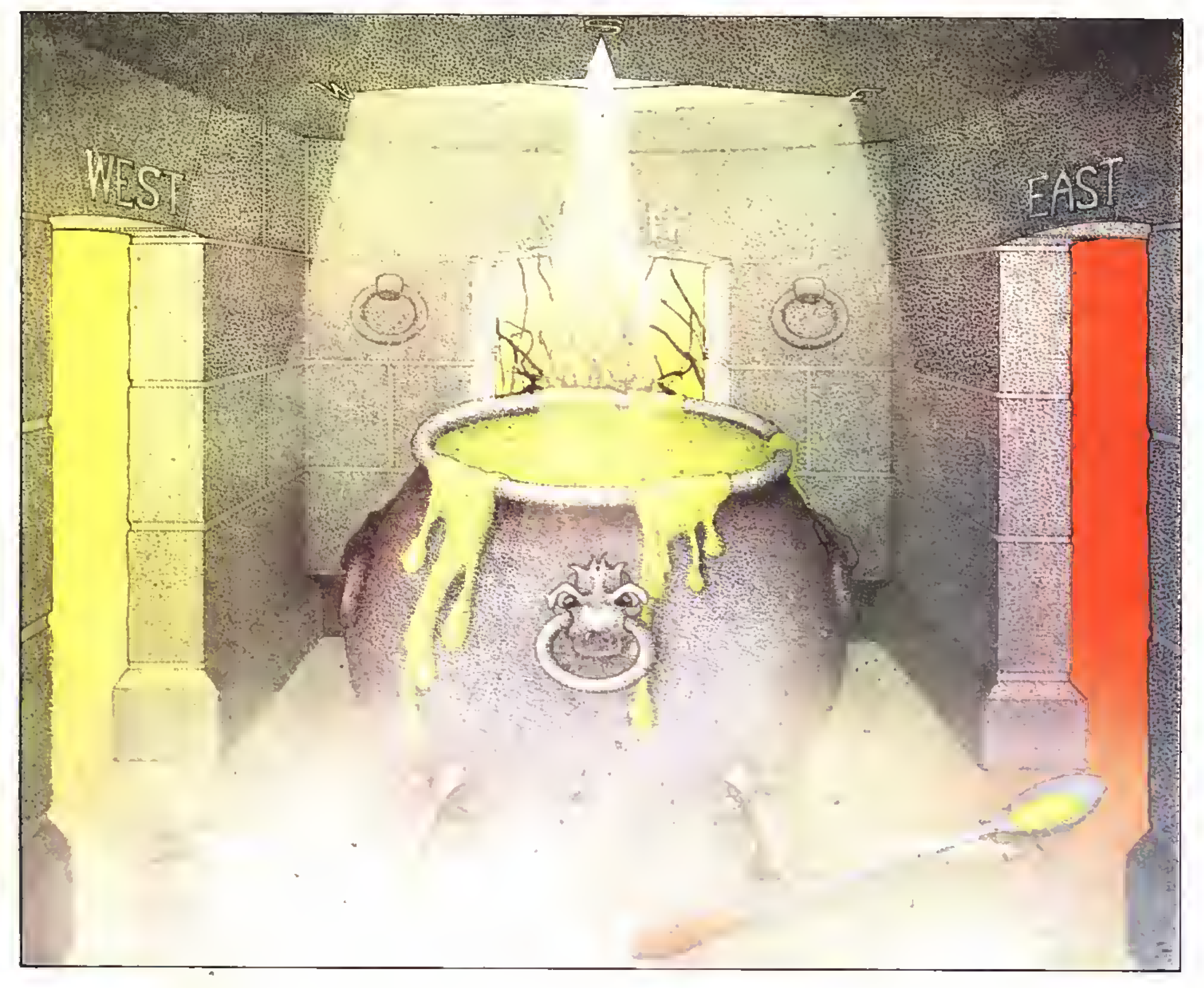
La mayoría de los objetos sólo se utilizan una vez en la aventura. Una excepción podría ser algo como una espada, que se puede utilizar muchas veces para luchar contra los malvados. Si tienes limitado el número de objetos que puedes acarrear, recuerda que lo más seguro normalmente es descartar los objetos una vez que los has utilizado.

Dibuja un mapa siempre. Marca sobre el mismo los nombres de las estancias, todo lo que sea de interés sobre cada una de ellas, los objetos que haya dentro, todas las entradas y salidas, y sus direcciones.

El mapa te ahorrará mucho tiempo y esfuerzo cuando tengas que volver sobre tus pasos, cosa que tendrás que hacer muchas veces durante el juego.

Si tienes que abandonar algún objeto debido a que no puedes llevarlo todo, no te olvides de marcar su posición en el mapa. Y lo que también es muy importante, el dibujo del mapa permitirá que te asegures de explorar toda la aventura, con lo que no tendrás que volver a nadar en las arenas movedizas por enésima vez.

Casi todos los juegos te permiten pedir un inventario de los objetos que llevas. De vez en cuando es una buena idea examinar qué objetos tienes exáctamente a mano, tecleando IN-



VENTARIO, INVE o simplemente I, dependiendo de la aventura de que se trate.

Algunas aventuras te permiten además pedir ayuda, también esto depende del juego, así como la forma de pedirla. Pudedes conseguir o no una sugerencia útil, lo más frecuente es que te encuentres con algo como AOUI NO HAY AYUDA.

Algunos juegos siguen muy de cerca la descripción de un libro particular, en cuyo caso el estudio del libro en cuestión es decididamente una buena idea. Otros juegos toman prestadas pequeñas secciones o ideas. Si crees reconocer algo y no puedes resolver un problema particular, intenta buscar en el libro. Análogamente si un extraño personaje con un cnorme hacha bloquea tu camino y te pregunta el diámetro de la Tierra, no lo dudes, !véte y averígualo!

Otro artificio muy usado en los juegos de aventuras son los equívocos. Míralos bien, no todas las cosas son lo que parecen.

También puede ser una buena idea mantener un directorio de sinónimos, y manejarlo para probar todas las variaciones posibles de una frase particular. Por ejemplo, el programador podría no haber incluído RESTREGAR además de FROTAR.

Y un último truco. Si la aventura que estás jugando te permite guardar (con SAVE) alguna parte, y estás a punto de intentar algo peligroso, guárda la situación en la que te encuentras, antes de probar. Si te matan, simplemente volverás a cargar el juego y podrás continuar desde donde estabas. Con este sistema tendrás muchos intentos para matar al dragón, atravesar un puente que se tambalea o escapar de una caverna.

ESCRIBIENDO AVENTURAS

Escribir aventuras es una buena manera de ponerse seriamente a estudiar BASIC. Se utilizan casi todos los aspectos importantes dcl lenguaje: manejo de cadenas, las distintas modalidades de PRINT para formateo de pantalla, variables, cadenas, etc.

La mayoría de las aventuras comerciales están aún escritas en BASIC, debido a que realmente no hay necesidad real de aprovechar la velocidad del código máquina. La única barrera para que produzcas juegos de una calidad absolutamente superior es tu propia imaginación.

Sin embargo, antes de sentarte a programar tu aventura, debes tener una idea muy clara de lo que vas a hacer. Si quieres ahorrarte muchos quebraderos de cabeza, debes planear por adelantado los dibujos, los enigmas, los peligros, etc.

Primero coge un papel y anota unas cuantas ideas. No te preocupes si no tienes una visión completa de todo lo que supone un juego de aventuras; lo que necesitas es una idea para una historia, un lugar para la aventura y unos cuantos rompecabezas para que los resuelva el jugador. A medida que vayamos avanzando en esta sección de nuestro coleccionable, verás cómo una idea sobre un juego se va convirtiéndo en una aventura y aprenderás a adaptar tus propias ideas originales.

Has de ser muy cuidadoso con el mundo que elijas. Intenta que sea lo más interesante posible, si pretendes que una aventura resulte muy apasionante en el interior de un bloque de oficinas, te va a costar lo tuyo.

Para tu inspiración puedes acudir a libros, películas, la televisión o cualquier otra posible fuente. También puedes sacar ideas de otros programas de aventuras, aunque probablemente la mejor fuente de inspiración es...!una mente ligeramente retorcida! Busca siempre un tema o idea central que puedas ir desarrollando por toda la aventura.

Intenta conseguir el adecuado equilibrio entre desafío e imposibilidad. No es bueno gastar mucho tiempo y esfuerzos escribiendo una aventura que cualquiera pueda resolver en media hora. Recíprocamente, no ganarás muchos amigos si tu aventura es totalmente imposible de resolver.

La regla de oro es «dar algunas posibilidades a los jugadores, !pero no demasiadas!»

Intenta no dejar muchas habitaciones vacías en tus aventuras. Realmente no añaden nada a la misma y ocupan un espacio de memoria que es vital. Además contribuyen a que la aventura sea más aburrida.

No hagas que tus primeras aventuras sean muy complejas, ya que los problemas que originen pueden ser muy difíciles de depurar hasta que adquieras cierta práctica. Aprende a conocer todo lo que interviene antes de intentar algo muy ambicioso. Ten a la vista cuánta memoria tiene aún disponible tu máquina.

En la aventura que verás construir más adelante hay muchas sentencias REM. Para ahorrar memoria en una gran aventura es mejor no poner muchas, pero al principio contribuyen a que resulte más fácil de escribir.

También se puede ahorrar memoria en las descripciones de los lugares. No las hagas demasiado cortas, porque podrías perder todo el sabor de la aventura. A ti te corresponde establecer el correcto equilibrio entre sabor y espacio.

En los próximos números veremos cómo convertir una idea sobre una aventura en un mapa y como empezar un programa.

¿CUANTA MEMORIA QUEDA?

Cuando estás escribiendo un gran juego de aventuras, es muy fácil que te encuentres con que has sobrepasado los límites de la memoria de tu máquina. Evidentemente, los problemas de pasarse de capacidad de memoria son más agudos en el caso de máquinas con menor memoria. Hay una forma de comprobar cuánta memoria queda, basta con que utilices la función FRE.

No tienes más que teclear:

```
PRINT FRE(0)
```

La memoria que queda tiene en cuenta tanto el espacio ocupado por el programa como el espacio ocupado por las variables. La mejor manera de encontrar la cantidad exacta de memoria restante es ejecutar el programa (con RUN), si es posible durantc el desarrollo.

PROYECTA TU AVENTURA

Al escribir una aventura, la primera etapa es perfilar un bosquejo general de la historia y dibujar un mapa tosco de todos los lugares que intervienen. Esto constituye la base de todo el programa.

Es esencial que tengas toda tu historia lista antes de empezar a programar. Si no lo haces así, es muy probable que tengas muchas dificultades, con muchos errores y cabos sueltos difíciles de atar.

Para ver cómo se hace esto dedicaremos los siguientes capítulos al desarrollo de un programa de aventuras típico (aunque necesariamente muy corto). La acción de la aventura se sitúa en algún lejano país donde el jugador tiene que conquistar el fabuloso ojo perdido de una purpúrea estatua. Si sigues todos los pasos al escribir esta aventura, rápidamente verás la forma de escribir las tuyas propias.

LA HISTORIA

Tienes que crear un mundo que se adapte a las líneas generales de la historia. Has de encontrar objetos adecuados y asignarles un papel, y además tienes que preparar enigmas para ser resueltos.

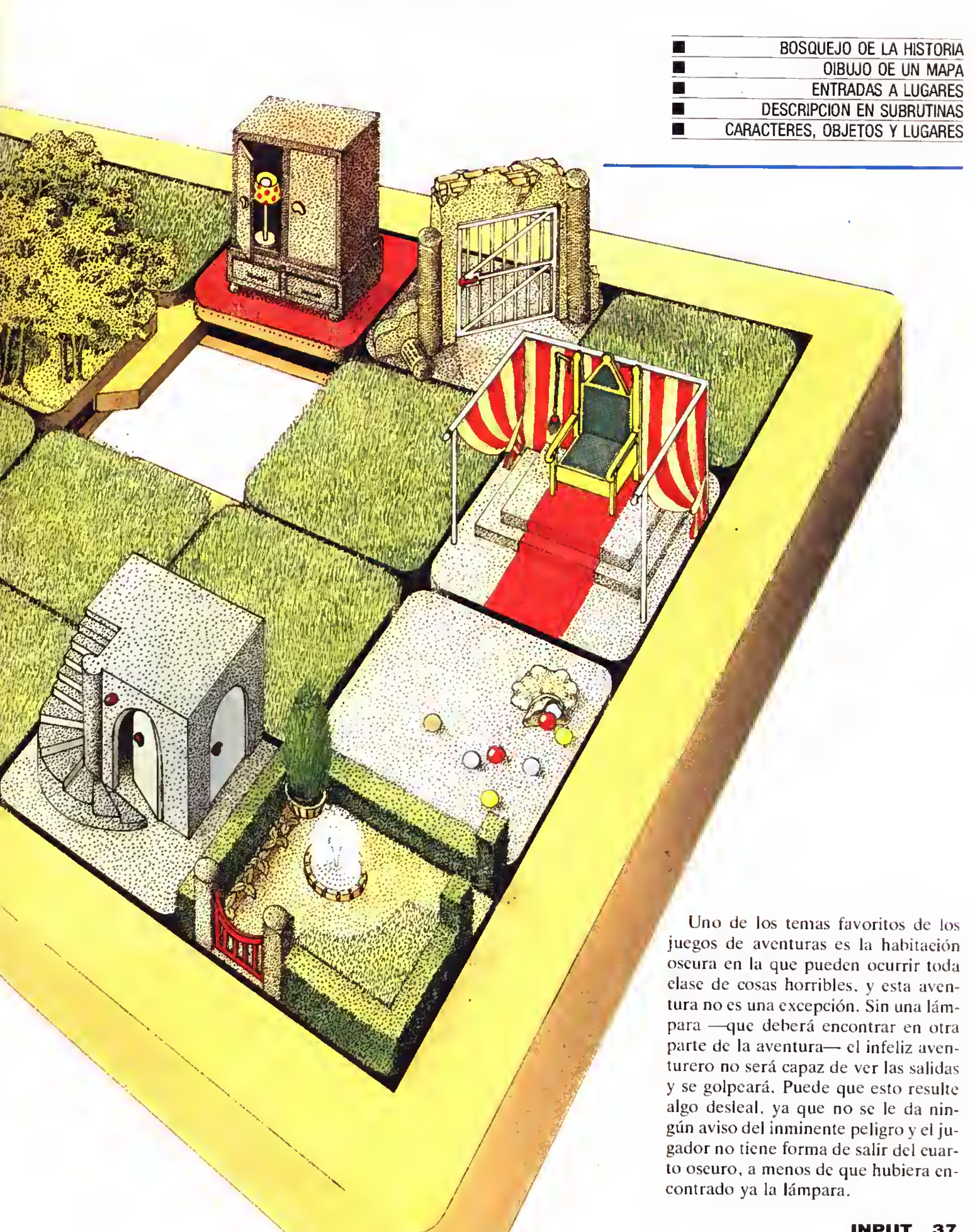
No hace falta que te ocupes de todo esto a la vez, ya que a medida que vas pensando la aventura, la historia va tomando cada vez una forma más definida y los detalles van encajando en su sitio. Empieza pues bosquejando la historia a grandes rasgos.

El jugador de la aventura atraviesa unas dificultades financieras espantosas, se ha embarcado en la búsqueda del fabuloso (y muy valioso) globo ocular que está escondido en alguna parte en el mundo de la aventura. Por desgracia la Delegación de Contribuciones ha enviado tras él a un inspector fiscal. El papel del inspector de impuestos en esta aventura es parecido al papel del pirata en otras, a saber, hacer estragos en el pobre aventurero. Si aparece el inspector de impuestos pueden ocurrir dos cosas. Si llevas algún objeto, lo cogerá para cobrarse tu enorme deuda tributaria, pero si todavía no has encontrado nada (y en consecuencia no puedes pagar), te encadenará en una sombría mazmorra.

Este es el esqueleto básico de la idea. Ahora tienes que ir cubriendo algunos detalles, como qué objetos se pueden encontrar durante la búsqueda. En nuestra aventura, hemos decidido que la regla general de recoger el mayor número posible de objetos no sea válida. Esta vez, no todos los objetos son provechosos para la búsqueda; de hecho uno de ellos no valdrá absolutamente para nada. O para casi nada, es un objeto pesado (por ejemplo un ladrillo) que te matará si intentas cruzar el río a nado.

El objeto más importante de todos es el globo ocular, y para añadir más interés, se ha previsto una manera de esconderlo o enmascararlo. Podría estar escondido dentro de un cofre o en una cámara acorazada, pero hemos elegido una forma mucho más astuta de despistar al aventurero. En vez de ocultar la joya en un sitio que obviamente contiene cosas de valor, estará camuflada en una bolsa de canicas. ¡Cualquier intento de jugar a las canicas, no conducirá al aventurero a ninguna parte!

- BOSQUEJO OE LA HISTORIA
- OIBUJO OE UN MAPA
- ENTRADAS A LUGARES
- DESCRIPCION EN SUBRUTINAS
- CARACTERES, OBJETOS Y LUGARES



Uno de los temas favoritos de los juegos de aventuras es la habitación oscura en la que pueden ocurrir toda clase de cosas horribles, y esta aventura no es una excepción. Sin una lámpara —que deberá encontrar en otra parte de la aventura— el infeliz aventurero no será capaz de ver las salidas y se golpeará. Puede que esto resulte algo desleal, ya que no se le da ningún aviso del inminente peligro y el jugador no tiene forma de salir del cuarto oscuro, a menos de que hubiera encontrado ya la lámpara.

Para neutralizar las acciones del inspector de impuestos, y dar al aventurero alguna probabilidad más, por alguna parte de la aventura se esconderá un arma, tal vez una pistola o un cuchillo.

Finalmente, por pura diversión, hemos puesto un salón del trono y una cadena. El salón del trono no es exactamente lo que parece ser. De hecho, si no llevas la joya, al tirar de una cadena, el agua te arrollará y serás violentamente expulsado de la aventura.

Oueda una cosa por establecer, la más importante de todas, las condiciones para ganar el juego. No hay una salida del mundo, y una parte del enigma es cómo escapar con la joya.

Evidentemente, para ganar el juego el aventurero tiene que haber encontrado la joya; no basta con la bolsa de canicas. Para que sea aún más difícil, el aventurero tiene que estar además en el salón del trono. Si tira de la cadena esta vez, no le arrastrará hacia dentro del inodoro.

La ventaja de que la vía de salida sea un peligro en otras condiciones, es que se desanime el jugador cauto y no intente entrar allí muy a menudo, lo cual prolonga el juego.

LA HISTORIA HASTA AHORA

Es el momento de hacer una recapitulación, antes de que pierdas las pistas de los temas, que empiezan a poner ya la cosa un poco complicada. Puede ser útil hacer una lista antes de empezar con el mapa. Para la aventura podría ser algo así:

PERSONAJES:
- Aventurero
- Inspector de hacienda; aparecerá aleatoriamente

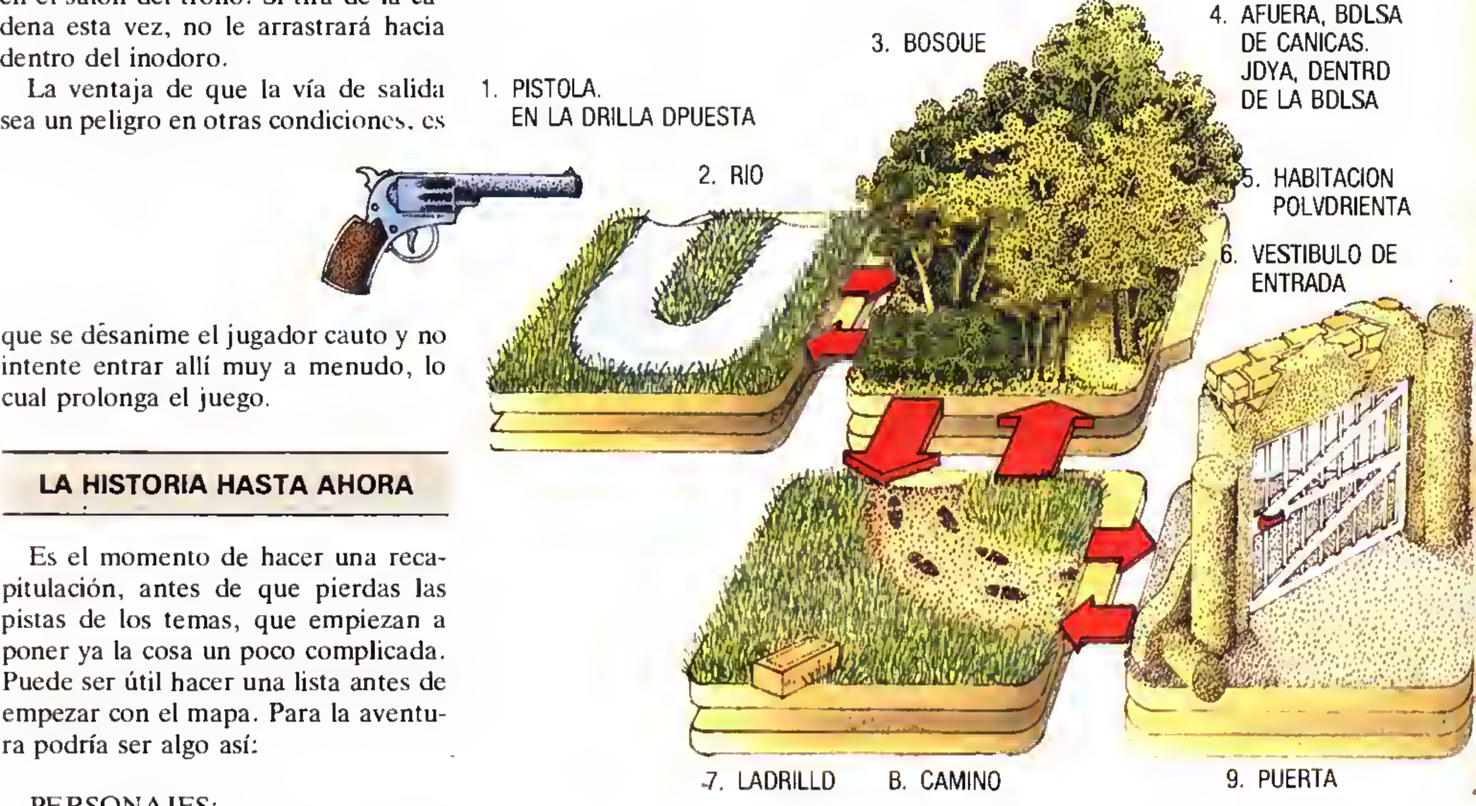
OBJETOS:
- Globo ocular; escondido en una holsa de canicas
- Ladrillo; motivo de distracción que matará al aventurero si intenta cruzar el río a nado
- Lámpara; necesaria para encontrar la salida de una habitación a oscuras
- Pistola; arma para matar al inspector de impuestos
- Cadena; en el salón del trono

LUGARES:
- Río
- Cuarto oscuro
- Salón del trono

Hasta ahora en la avenura sólo se han fijado tres de los lugares por las cosas que tiene que ocurrir en ellos. En este punto podrías haber decidido algunas cosas más. Pero en cualquier caso, tu siguiente paso es reunir todos estos temas en un plano del mundo de la aventura.

EMPEZANDO CON EL MAPA

Probablemente tu primer mapa consistirá simplemente en una serie de cajas conectadas por medio de flechas, como en la figura 1. Cada una de las cajas representa una habitación o un lugar del mundo; lugar es probablemente el término más adecuado ya que las aventuras no están limitadas a interiores, y lugar puede ser cualquier cosa, desde una cabeza de alfiler en el dobladillo de la reina hasta una enorme llanura que se extiende hasta donde alcance tu vista. Tienes que incorporar todos los lugares en tu lista pre-



El primer mapa de la aventura muestra todos los lugares proyectados y sus posiciones relativas. Las flechas indican salidas que están siempre abiertas, las flechas con rayas indican salidas que sólo se pueden utilizar bajo especiales condiciones; en este caso, cuando se ha encendido la lámpara.

liminar, más otros que te permitan enlazar el juego.

Al dibujar este mapa, acuérdate de marcar los sentidos en que se puede circular en cada habitación, porque puede ser que haya puertas que quieras que sólo puedan cruzarse en una dirección, acompañadas de un mensaje tal como éste:

LA PUERTA SE CIERRA DE GOLPE JUSTO DETRAS DE TI

Las líneas de trazos que hay junto al cuarto oscuro indican que el aventurero sólo podrá ir en esa dirección si se cumplen ciertas condiciones. En este caso, la condición es que el aventurero tenga la lámpara y la haya encendido para poder ver las salidas.

Es muy difícil predecir cuántos lugares se pueden meter en una cantidad dada de RAM. La dificultad surge porque hay muchas cosas que se disputan el espacio de memoria en el programa de la aventura: descripciones de lugares, palabras que quieres que el programa reconozca, número de objetos y lo que quieres que se haga con ellos, número de enigmas y su complejidad, etc.

Cuando hayas escrito unas cuantas aventuras pequeñas y comprobado cuánta memoria ocupan, tendrás una idea de lo que puedes meter en tu máquina.

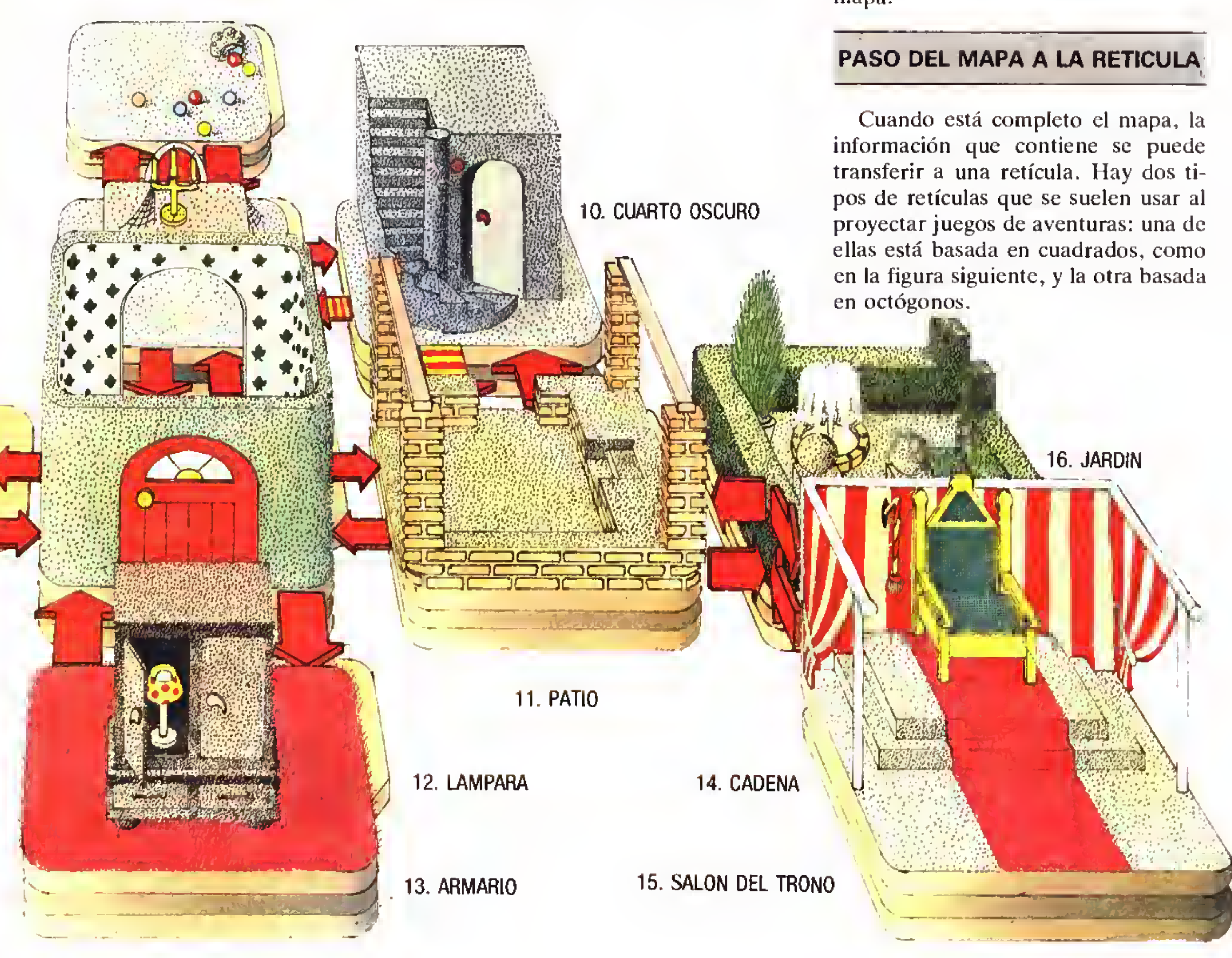
Los poseedores de un ordenador de 16K pronto descubrirán que es casi imposible escribir una aventura en gran escala en una cantidad tan pequeña de RAM. Sin embargo, la aventura que desarrollaremos en los siguientes capítulos sólo tiene unos pocos lugares —12 en total— por lo que es suficientemente pequeña para no causarte dolores de cabeza.

El mapa para la búsqueda de la joya del globo ocular podría ser algo parecido al de la figura. En esta fase las descripciones de los lugares han de mantenerse cortas. Se han dibujado los enlaces y se ha decidido el punto de partida. Esto es muy importante, ya que afecta a la manera en que se aborda la aventura, al orden en que aparecen los objetos y se plantean los *puzzles*.

También están marcados los objetos en sus lugares. Los objetos que aparecerán más tarde, por ejemplo la joya, es mejor listarlos a un lado del mapa.

PASO DEL MAPA A LA RETICULA

Cuando está completo el mapa, la información que contiene se puede transferir a una retícula. Hay dos tipos de retículas que se suelen usar al proyectar juegos de aventuras: una de ellas está basada en cuadrados, como en la figura siguiente, y la otra basada en octógonos.



En las aventuras más sencillas, sólo se podrá salir de cada lugar en una de estas cuatro direciones: Norte, Sur, Este y Oeste (así ocurre en la aventura del globo ocular). Si tu aventura es de este tipo, tienes que transferir a la rejilla cuadrada toda la información para que resulte equivalente al mapa. Más adelante desarrollaremos esto con detalle. Si has utilizado salidas que incluyen Nordeste, Noroeste, Sureste y Suroeste, tienes que emplear la retícula octogonal, aunque este tipo de rejillas resulta complicado.

La última variante de la rejilla se produce si has decidido también subir y bajar. En este caso la mejor solución es utilizar una rejilla separada para cada «nivel» de aventura.

La aventura del ojo precioso está basada sobre una retícula cuadrada, permitiendo sólo salidas por el Norte, Sur, Este y Oeste. A menos que haya una necesidad real de otras direcciones, este tipo de aventuras resulta muy adecuado y hay una manera de obviar las subidas y bajadas. Si utilizas una descripción con frases como

HAY UNA ESCALERA QUE BAJA HACIA EL OESTE

puedes usar la respuesta normal del Oeste o la rutina adecuada.

LA RETICULA

Esta aventura requiere una retícula cuadrada de 6×4 como puedes comprobar contando el número máximo de lugares de tu mapa, de arriba a abajo y de derecha a izquierda. Antes de empezar a trasladar todos los detalles a la retícula, asegúrate de que has numerado todos los cuadrados. Empieza con el número 1 en la parte superior izquierda, y prosigue hasta llegar a la parte inferior derecha. Cuando hayas numerado todos los cuadrados y transferido todos los detalles, la retícula será como la de la figura.

EMPIEZA EL PROGRAMA

Ahora tienes una línea histórica y una retícula completa, puedes empezar con el programa.

El primer paso es teclear las descripciones de los lugares, basándote en tu retícula. Tienes que decidir cómo van a ser de largas. Intenta sugerir lo más posible el ambiente de la aventura sin derrochar memoria. Aparte de las descripciones de los lugares, hay que decirle al ordenador en qué direcciones se encuentran las salidas.

Aquí tienes al fin las primeras secciones del programa. Los números de líneas tan altos te permitirán disponer de suficiente espacio en las anteriores secciones de programa a medida que vayas desarrollando el juego.

Teclea esta sección y almacénala (SAVE) en cinta:

```
5000 REM ** DESCRIPCION DE
     UBICACIONES
5010 REM ** UBICACION 4
5020 PRINT"ESTAS EN EL
     EXTERIOR DE UN GRAN
     EDIFICIO"
5030 N=0:E=0:S=1:O=0:RETURN
5040 REM ** UBICACION 7
5050 PRINT"ESTAS EN UN RIO
     TURBULENTO"
5060 N=0:E=1:S=0:O=0:RETURN
5070 REM ** UBICACION 8
5080 PRINT"ESTAS EN UN BOSQUE
     PETRIFICADO"
5090 N=0:E=0:S=1:O=1:RETURN
5100 REM ** UBICACION 10
5110 PRINT"ESTAS EN UNA SALA
     POLVORIENTA"
5120 N=1:E=1:S=1:O=0:RETURN
5130 REM ** UBICACION 11
5140 PRINT"ESTAS EN UN CUARTO
     OSCURO"
5150 IF OB(6)<>-1 OR LA<>1
     THEN N=0:E=0:S40:O=0
5155 IF OB(6)<>-1 OR LA<>1
     THEN PRINT"NO SE VEN LAS
     SALIDAS EN LA OSCURIDAD"
     :RETURN
5160 N=0:E=0:S=1:O=1:RETURN
5170 REM ** UBICACION 14
5180 PRINT"ESTAS EN UN CAMINO
     LLENO DE BARRO"
5190 N=1:E=1:S=0:O=0:RETURN
5200 REM ** UBICACION 15
5210 PRINT"ESTAS EN LA PUERTA
     DE LA CIUDAD OCULTA"
5220 N=0:E=1:S=0:O=1:RETURN
5230 REM ** UBICACION 16
5240 PRINT"ESTAS EN EL HALL
     DE ENTRADA"
5250 N=1:E=1:S=1:O=1:RETURN
5260 REM ** UBICACION 17
5270 PRINT"ESTAS EN EL
     PATIO"
5280 N=1:E=1:S=0:O=1:RETURN
5290 REM ** UBICACION 18
5300 PRINT"ESTAS EN EL
     JARDIN"
5310 N=0:E=0:S=1:O=1:RETURN
5320 REM ** UBICACION 22
5330 PRINT"ESTAS EN EL
     ARMARIO"
5340 N=1:E=0:S=0:O=0:RETURN
5350 REM ** UBICACION 24
5360 PRINT"ESTAS EN LA SALA
     DEL TRONO"
5370 N=1:E=0:S=0:O=0:
     RETURN
```

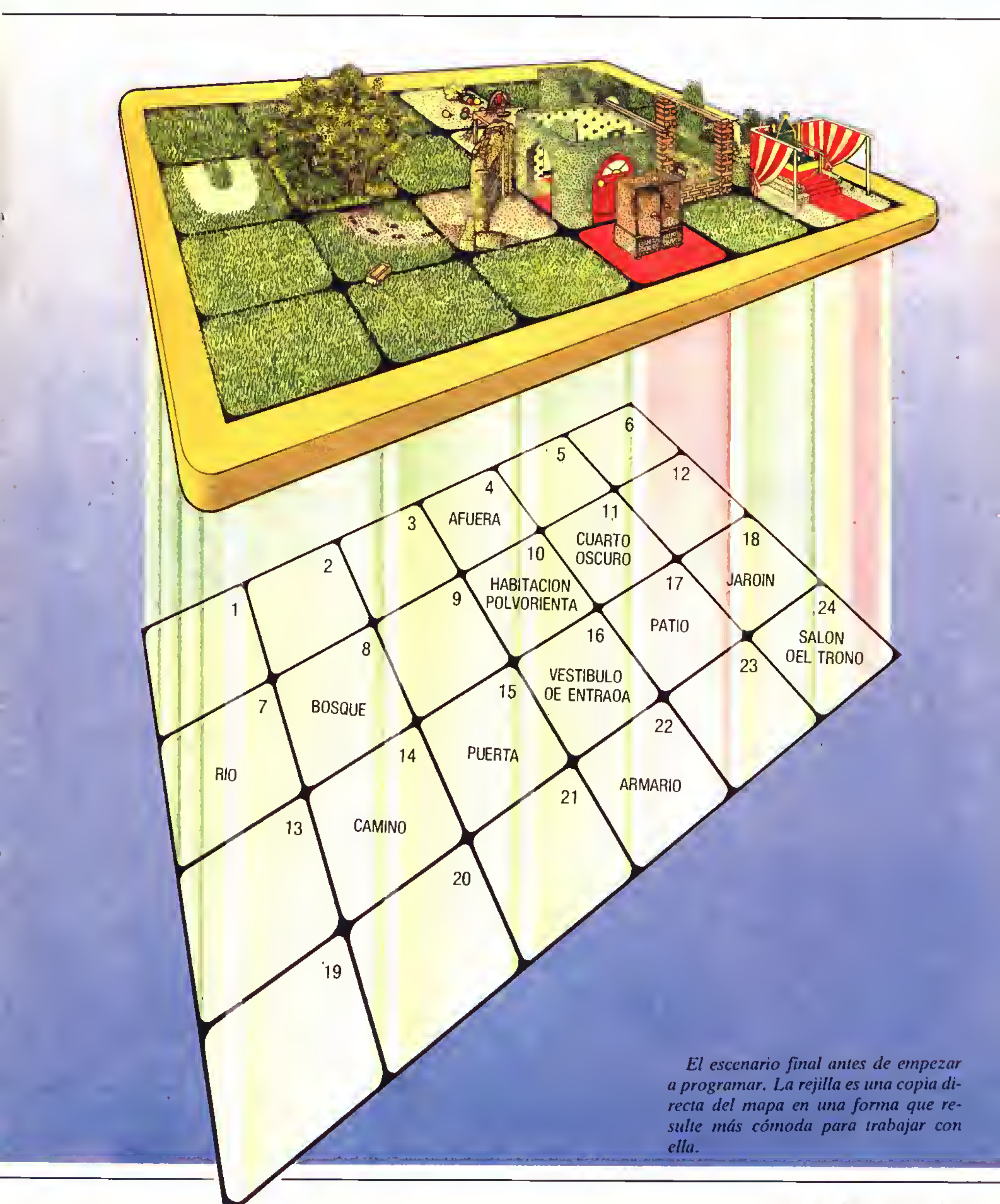
No te preocupes por el abundante uso de las sentencias REM y toda la memoria que consumen. En esta primera etapa del desarrollo del programa es muy importante saber qué es lo que hace cada parte del mismo, o a qué número de lugar se refiere una descripción particular. Siempre puedes quitarlas más adelante.

Después de cada línea con la descripción de un lugar, hay otra línea que contiene información sobre sus posibles salidas. Las variables N, S, E y O significan Norte, Sur, Este y Oeste. Estas variables pueden tomar uno de dos valores posibles: 0 significa que no hay salida en esa dirección, mientras que 1 significa que hay salida.

Finalmente, después de cada sección de este programa hay un RETURN, debido a que cada descripción de lugar se llama mediante una sentencia GOSUB.

La sentencia IF...THEN que hay en la sección del cuarto oscuro es para comprobar si el aventurero tiene la lámpara, pero la descripción de las variables se comentará más adelante cuando nos ocupemos de los objetos.

En el próximo capítulo veremos la manera en que se mueve el aventurero por los distintos lugares.

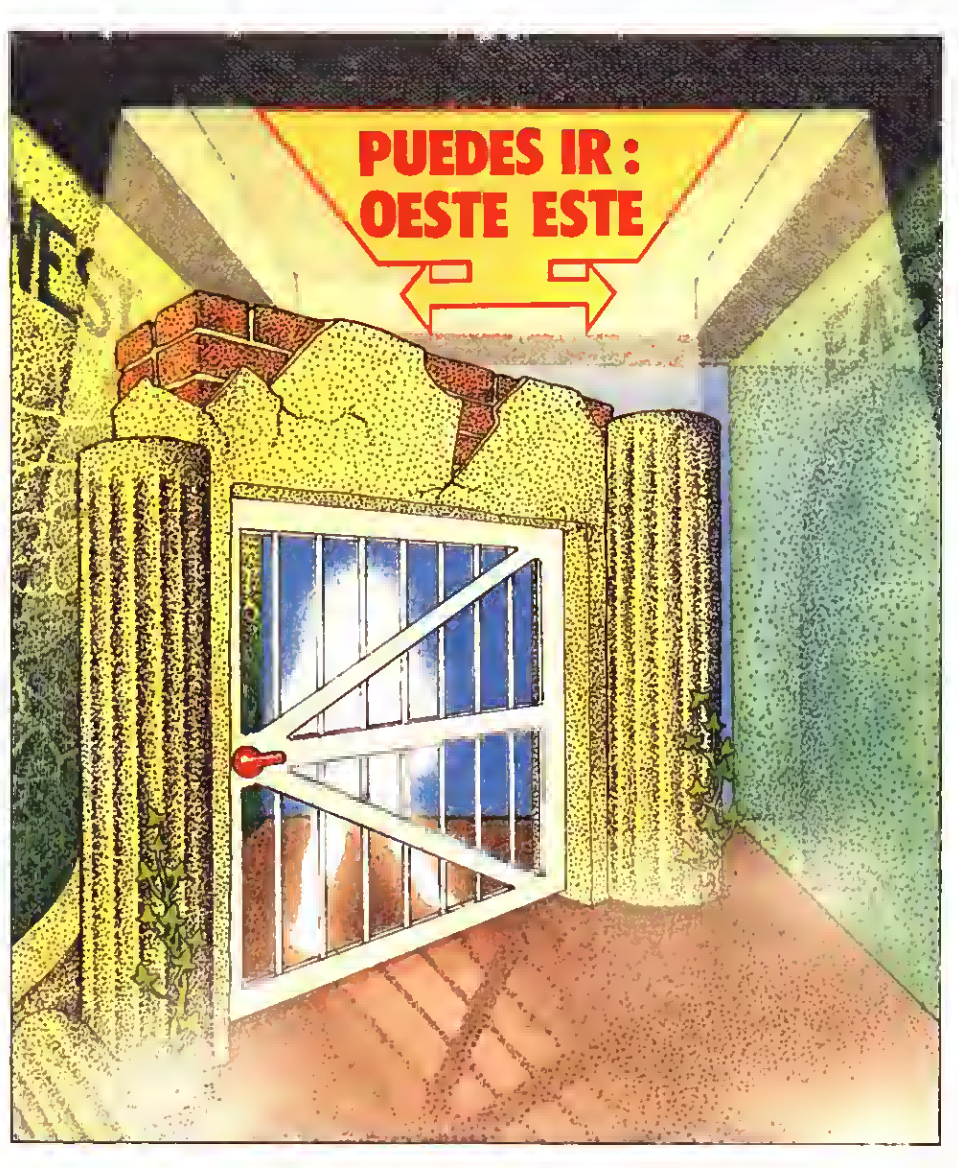


El escenario final antes de empezar a programar. La rejilla es una copia directa del mapa en una forma que resulte más cómoda para trabajar con ella.

UNA AVENTURA MOVIDA

- PRESENTACION DE DIRECCIONES
- TRATAMIENTO DE LAS INSTRUCCIONES
- MOVIMIENTO A TRAVES DE LA AVENTURA
- BLOQUEO DE LOS MOVIMIENTOS IMPOSIBLES

Una parte de la diversión de jugar juegos de aventuras es la posiblidad de explorar mundos nuevos y extraños sin salir de casa. Veamos la manera de iniciar estas exploraciones.



Ahora que ya has tecleado un programa que contiene todas las descripciones de los ambientes, es el momento de hacer que el aventurero pueda explorarlos, desplazándose de un lugar a otro. Tienes que prever todos los movimientos posibles en cada punto, y fijar unos límites, basándote en el grado de dificultad y en las pistas que se van recogiendo a medida que progresa la aventura.

En esta ocasión, para ampliar el desarrollo de tu programa de aventuras, presentaremos rutinas con la descripción correcta de los ambientes, junto con las posibles salidas a los mismos. Al jugador se le pedirá que introduzca respuestas, y verás la manera de escribir una seccion de programa para que el jugador se vaya moviendo por el interior del fantástico mundo, en base a las elecciones hechas, a medida que progresa la aventura.

EL PUNTO DE PARTIDA

Lo primero que el ordenador tiene que saber es dónde está el aventurero en cada momento del juego. Para hacer esto el programa mira al valor que va tomando una variable llamada L (de lugar). A esta variable se le asigna un valor que corresponde al ambiente en que se encuentra el aventurero después de cada movimiento.

Para empezar pues la aventura, tienes que decirle al ordenador dónde quieres que se encuentre el aventurero en el momento de empezar.

Aquí tienes la primera parte de un programa que se encarga de esto. Carga (con LOAD) la sección que tecleaste la última vez, y añádele las siguientes líneas:

```
270 REM ** POSICION DE
    COMIENZO
280 L=15
290 GOTO 330
```

El número 15 corresponde al lugar en que se encuentra la puerta de la ciudad escondida. Si quieres que la aventura empiece en otro lugar, basta con que modifiques el valor de L. Enseguida veremos la forma de modificar el valor de L durante el juego, según el lugar atravesado. Pero antes de que el aventurero pueda empezar a moverse, hay que decirle al ordenador hacia dónde tiene que ir. Para ello el aventurero utilizará el teclado, a través del que tendrá que introducir una o varias palabras.

RESPUESTAS

Para que el ordenador pueda entender adecuadamente tus respuestas, y actuar en consecuencia, tienes que proporcionarle una lista de todas las palabras que puede reconocer.

En esta etapa del desarrollo, basta con que reconozca las cuatro direcciones NORTE, SUR, ESTE y OESTE. Podemos meterlas en una matriz R$, que contiene los datos de cada dirección de respuesta.

```
110 REM ** MATRICES DE
    RESPUESTAS
120 DIM R$(19),R(19):KEY OFF:
    WIDTH 37:CLS
130 FOR K=1 TO 4:READ R$(K)
    ,R(K):NEXT
150 DATA NORTE,1,SUR,1,ESTE,1
    ,OESTE,1
```

En la línea 120 se dimensionan las matrices, de modo que contengan todas las respuestas necesarias para el juego, dado que por el momento sólo necesitas utilizar las direcciones, sólo se emplearán los cuatro primeros elementos de las matrices R$ y R. El bucle FOR ... NEXT de la línea 130, va desde uno hasta cuatro, leyendo tanto en R$ como en R. Las direcciones y sus números están en las sentencias DATA de la línea 150.

Pero evidentemente esta información no le sirve para nada al jugador, a menos que el ordenador le diga previamente dónde se encuentra.

LA BUSQUEDA DE UN LUGAR

Para que el aventurero pueda saber adonde ha ido a parar después de cada uno de sus movimientos, hay que suministrarle una descripción de los lugares. Estas descripciones de lugares ya las tienes tecleadas; sólo te hace falta una rutina para extraer la descripción que corresponda al valor de la variable L, el número de lugar. Aquí es donde resultan útiles las líneas REM que introdujiste anteriormente.



```
300 REM ** ENCONTRAR
    UBICACION
310 CLS
330 IF L<11 THEN ON L GOSUB 0
    ,0,0,5020,0,0,5050,5080,0
    ,5110:
GOTO 400
340 IF L<21 THEN ON L-10
    GOSUB 5140,0,0,5180,5210,
    5240,5270,5300,0,0
    :GOTO 400
350 IF L<26 THEN ON L-20
    GOSUB 0,5330,0,5360
```



Antes de escribir este tipo de rutinas, tienes que asegurarte del número correspondiente a cada descripción de lugar. A partir del lugar número 1, tienes que ir haciendo una lista de los números de líneas de cada descripción. Si para alguno de los ambientes no hay ninguna descripción especial, asígnale un 0. En nuestra aventura, por ejemplo, no hay descripciones para los lugares 1, 2 y 3, pero sí la hay para el lugar 4.

Ahora que tienes la lista de los números de líneas, puedes empezar a escribir la rutina. Es una sencilla secuencia de operaciones de control del valor de L por medio de una sentencia ON ... GOSUB.

La lista de números de líneas va desde la línea 330 a la 350, comenzando por el lugar 1 al principio de la línea 330 y terminando con el lugar 24 al final de la línea 350.

PRESENTACION DE LAS DIRECCIONES

Aparte de conocer las descripciones de los lugares, el jugador de la aventura querrá saber qué salidas tiene. Como en cada lugar no le será posible moverse en todas direcciones, el programa debe tener alguna forma de controlar las informaciones asociadas

con la dirección, es decir las variables N, S, E y O. La siguiente rutina dirá al aventurero qué direcciones son posibles:

```
390 REM ** MUESTRA
DIRECCION
400 IF L<>11 OR (LA=1 AND
    OB(6)=-1) THEN PRINT
    :PRINT"PUEDES IR ";:GOTO
    410
405 GOTO 460
410 IF N>0 THEN PRINT TAB(11)
    "NORTE"
420 IF S>0 THEN PRINT TAB(11)
    "SUR"
430 IF E>0 THEN PRINT TAB(11)
    "ESTE"
440 IF O>0 THEN PRINT TAB(11)
    "OESTE"
```

La rutina se limita a comprobar el valor de las variables N, S, E y O, basándose en el mapa de direcciones. Si el valor de las variables es mayor que cero, la dirección es posible y se presenta la correspondiente salida.

Esta rutina puede ser incorporada tal como está en cualquier aventura basada en una retícula de cuadrados.

INSTRUCCIONES

Ahora que el aventurero ya conoce las direcciones que puede tomar, hay que bacerle algunas sugerencias. La siguiente sección de programa le preguntará al jugador ¿QUE VAS A HACER AHORA?

```
450 REM ** INSTRUCCIONES
460 PRINT:INPUT"AHORA
QUE";I$
470 GOSUB 3010
```



Se trata de una rutina de entrada muy sencilla. A la respuesta suminis-

ESTAS EN LA ENTRADA

trada la llamamos I$. El ordenador debe comprobar que la respuesta es correcta y actuar en consecuencia. La línea 470 envía el control a la subrutina de la línea 3010, que es la encargada de procesar la respuesta dada por el jugador.

```
3000 REM ** COMPRUEBA
     INSTRUCCION
3010 N$="":FOR Z=1 TO LEN(I$)
     :IF MID$(I$,Z,1)=" "
     THEN I=Z:GOTO 3020
3015 NEXT:I=0
3020 IF I=0 THEN V$=I$
     :GOTO 3050
3030 V$=LEFT$(I$,I-1)
3040 N$=MID$(I$,I+1)
3050 I=0
3060 FOR K=1 TO 19
3070 IF V$=LEFT$(R$(K),
     LEN(V$)) THEN I=R(K)
     :I$=LEFT$(V$,1)
3080 NEXT
3090 RETURN
```

Esta rutina comprueba si I$ contiene dos palabras. De ser así, llama V$ a la primera y N$ a la segunda. Hemos puesto V$ por verbo, como COGER, MATAR, o LLEVAR, y también sirve para las direcciones como NORTE, SUR, ESTE Y OESTE. N$ significa nombre, y vale para designar los objetos que aparecen en la aventura. El programa utiliza la sentencia MID$ (líneas 3010 a 3015) para examinar si hay un espacio entre V$ y N$.

Si se encuentra un espacio, la línea 3020 designa a las dos partes de I$ con las etiquetas N$ y V$. Si no se encuentra espacio, la propia línea 3020 llama V$ a toda I$.

El resto de la subrutina (líneas 3060 a 3080) compara las respuestas obtenidas con las de la matriz R$, que contiene las direcciones de las respuestas. Más adelante veremos cómo puede ampliarse para contener una serie adicional de verbos. Si en la línea 3070 se detecta una coincidencia, se pone en I el correspondiente valor de R. En fases posteriores del programa la máquina puede saber si ha habido una coincidencia examinando si I es mayor que cero. La última parte de la línea 3070, toma la primera letra de V$ y la llama I$, lo que servirá después para el movimiento del aventurero.

Esta subrutina puede utilizarse casi sin limitaciones en cualquier juego de aventuras. Probablemente la única modificación que le hagas, estará relacionada con la longitud del bucle FOR ... NEXT de la línea 3060.

P y R

¿Entenderá el programa respuestas de dirección tales como NORTE o IR NORTE, además de letras solas, tal como N?

La rutina de comprobación de instrucciones, situada en las líneas 3.000 a 3.090, está escrita de forma que admite palabras de cualquier número de letras. Cuando dos palabras van separadas por un espacio el programa las maneja de forma independiente.

La línea 3.070 toma, como respuesta de dirección, la primera letra. Siempre que esta primera letra sea N, S. E u O (en mayúscula), el programa entenderá lo que quiere decir el jugador, independientemente de cual sea la respuesta completa. Así pues: N, No y NORTE son respuestas válidas, pero no lo es IR NORTE.

Sin embargo, no hay nada que te impida hacer adiciones que permitan al aventurero utilizar respuestas del tipo IR NORTE. Más adelante veremos cómo maneja el programa los verbos y los nombres.

Lo que habrá que hacer será añadir IR a la lista de verbos y escribir una rutina de procesamiento de esta nueva forma verbal.

MOVIMIENTO

Para que el aventurero pueda explorar todos los lugares, basta que añadas al programa una rutina que transforme la variable de lugar L, según el contenido de I$. Aquí la tienes:

```
1000 REM ** RUTINA DE
     MOVIMIENTO
1010 IF I$="N" AND N>0 THEN
     L=L-6:GOTO 310
1020 IF I$="E" AND E>0 THEN
     L=L+1:GOTO 310
1030 IF I$="S" AND S>0 THEN
     L=L+6:GOTO 310
1040 IF I$="O" AND O>0 THEN
     L=L-1:GOTO 310
1050 REM ** SI NO HAY
     UBICACION POSIBLE EN TAL
     DIRECCION
1060 PRINT:PRINT"LO SIENTO"
     :PRINT"NO PUEDES IR EN
     ESA DIRECCION!"
     :GOTO 330
```

Recuerda que la aventura estaba basada en una retícula de seis por cuatro. El movimiento implica un cambio del valor de L, que depende del tamaño de la retícula. Moverse hacia el Norte o hacia el Sur equivale a sumar o restar 6 al valor de L para desplazarse una línea hacia arriba o hacia abajo en la retícula. Análogamente, el movimiento hacia el Este o hacia el Oeste es equivalente a sumarle o restarle 1 al valor de L.

Las líneas 1010 a 1040 comprueban las direcciones en I$ y ajustan L en consecuencia. Sólo es posible el movimiento si hay una salida que coincide con I$. Las salidas fueron definidas en las líneas inmediatamente siguientes a las descripciones de los lugares.

Si no existe salida en la dirección en que el aventurero quiere salir, la línea 60 presenta el mensaje ¡LO SIENTO! ¡NO PUEDES SALIR POR AHI!

Para utilizar esta rutina con una aventura diferente, el único cambio que tendrás que hacer tendrá que ver con el tamaño de la retícula. En tal caso, habrá que modificar las líneas 1010 a 1030, con arreglo a las líneas de la retícula.

Almacena ahora el programa (con SAVE) dejándolo listo para ser utilizado en el próximo capítulo.

LOS OBJETOS DE LA AVENTURA

- OATA PARA LOS OBJETOS
- OESCRIPCIONES LARGAS Y CORTAS
- MAS VERBOS
- TOMANOO Y OEJANOO OBJETOS
- RUTINA DE INVENTARIO

Ha llegado el momento de llenar tu vacío mundo de aventura con objetos. Te enseñaremos cómo puedes incorporar en el programa tu lista de objetos y la forma de manejarlos.



Al final del capítulo anterior teníamos un conjunto completo de ambientes para la aventura y ya habíamos proporcionado al aventurero la capacidad de moverse por todo el mundo de la aventura. Sin embargo en la fase actual las actividades del aventurero todavía carecen de sentido, ya que aún no sucede nada en ninguno de los lugares. Ha llegado el momento de volver atrás y ver lo que habías planeado incluir en cada punto.

Vamos a ver a continuación, la forma de agregar las rutinas necesarias para llegar al sitio adecuado y recoger o abandonar todos los objetos que intervienen en la aventura. También presentaremos una rutina que hace un inventario de todos los objetos que el aventurero lleva consigo en un momento dado, la cual puede resultar útil en los momentos más críticos de la acción.

Carga el programa que tienes desde el capítulo anterior y prepárate a introducir nuevas rutinas.

OBJETOS

La máquina necesita saber tres cosas acerca de los objetos de una aventura: el número del lugar donde el objeto estaba inicialmente situado, listo para que el aventurero lo encontrara, un nombre para el objeto y una descripción más larga que incluirá algo acerca de la situación del objeto y sugerirá de alguna forma su empleo. Estas tres cosas son indispensables para el ordenador, ya que en primer lugar tiene que saber si un determinado objeto está o no en un ambiente, además

tiene que poder avisar al jugador de su presencia, con ayuda de la descripción larga y, por último, necesita un nombre corto, para utilizarlo en instrucciones e inventarios.

Los números de los lugares se situarán en una matriz, el título de los objetos en otra y las descripciones de los objetos en una tercera. El programa procesará las tres matrices en paralelo; cada elemento de la matriz soporta una información acerca de los objetos, el primero es el número del lugar, el segundo es el nombre, etc.

Añade a tu programa las siguientes líneas:

```
160 REM ** MATRICES OBJETOS
170 READ NB
180 DIM OB(NB),OB$(NB),
    SI$(NB)
190 FOR I=1 TO NB:READ
    OB(I), OB$(I),SI$(I):
    NEXT
200 DATA 7,4,BOLSA,HAY UNA
    BOLSA DE CANICAS
210 DATA 14,LADRILLO,HAY UN
    LADRILLO A TUS PIES
220 DATA 24,CADENA,UNA CADENA
    CUELGA DEL TECHO
230 DATA 0,PISTOLA,HAY UNA
    PISTOLA EN EL SUELO
240 DATA 0,GLOBO OCULAR,EN EL
    SUELO HAY UNA ALHAJA CON
    FORMA DE OJO
250 DATA 22,LAMPARA,VES UNA
    LAMPARA
260 DATA 0,INSPECTOR FISCAL,
    UN INSPECTOR FISCAL
    APARECE DE REPENTE
```

Cada línea desde la 200 a la 260 contiene tres elementos de los DATA referidos al mismo objeto. La línea 200 incluye un elemento suplementario: el número 7 al principio de los datos informa a la máquina de cuantos conjuntos de datos hay.

Una vez que se lee el número 7 en la línea 170, se dimensionan tres matrices con ese tamaño en la línea 180. OB contendrá la situación de cada objeto, ya sea un número de lugar, un 0 si el objeto todavía no existe (por ejemplo el famoso ojo de brillantes que tiene que ser descubierto durante la aventura), o un −1 si se trata de un objeto que el aventurero lleva consigo. OB$ contendrá las descripciones cortas y S1$ las descripciones largas.

La línea 190 llena las matrices con datos de las líneas 200 a 260. Los datos se disponen en grupos de tres de la forma siguiente: número de lugar, descripción corta del objeto y descripción larga del mismo.

Cuando uses esta rutina para otras aventuras, no tendrás que hacer muchas modificaciones en su estructura, ya que al ajustar el primer elemento de los DATA, automáticamente se di-



mensionarán tanto el bucle FOR ... NEXT como las matrices.

DISPOSICION DE LOS OBJETOS

A continuación el programa contiene toda la información referente a la naturaleza y colocación de los objetos. La siguiente rutina presenta la descripción larga del objeto en el lugar adecuado:

```
360 REM ** VISUALIZACION DE
    OBJETOS EN EL LUGAR
    APROPIADO
370 FOR I=1 TO NB:IF
    OB(I)=L THEN PRINT
    SI$(I)
380 NEXT I
```



En esta fase tienes que hacer una pequeña modificación en las líneas 330 y 340: cambia el GOTO 400 por GOTO 370. Las líneas 370 y 380 examinan la matriz que contiene los lugares de los objetos. Si alguno de los números de los lugares coincide con el del lugar donde se encuentra el objeto en ese momento —variable L— aparece la descripción larga del objeto a continuación de la del lugar. Esta rutina se puede usar sin modificaciones en otras aventuras.

MAS VERBOS

Ya tienes en tu aventura unos cuantos objetos esparcidos por los diversos lugares, pero como la máquina todavía no entiende más palabras que NORTE, SUR, ESTE y OESTE, el pobre aventurero no puede hacer nada con esos objetos. Imagínate la frustración de no poder coger esa apetitosa bolsa de canicas o no poder defenderte contra el inspector de hacienda. Por eso tienes que darle al ordenador un vocabulario de palabras que pueda reconocer, diciéndole qué debe hacer con los objetos. Más adelante veremos qué hacer si el jugador introduce una palabra que no esté en el vocabulario suministrado a la máquina.

Ya que el programa trata todas las palabras de las direcciones como verbos, el mejor sitio para los verbos que indican lo que hacer con los objetos será la matriz R$, y el mejor sitio para los correspondientes números será la variable R.

En consecuencia tendrás que hacer algunas modificaciones, empezando por la línea 130. Tienes que cambiar los límites del bucle FOR ... NEXT. La nueva versión de dicha línea es:

```
130 FOR K=1 TO 19:READ R$(K)
    ,R(K):NEXT
```

Seguidamente, añade las líneas 140 y 145:

```
140 DATA NADO,5,VACIO,6,LUZ,7
    ,FIN,8,LISTA,9,MATO,10,
    DISPARO,10,AYUDA,11
145 DATA COJO,2,RECOJO,2,
    LLEVO,2,PONGO,3,DEJO,3,
    ABANDONO,3,TIRO,4
```

Cada verbo tiene su correspondiente número. Verbos con el mismo número tienen el mismo significado, por lo que al ordenador se refiere, y realizarán la misma operación. Por ejemplo, programando las cosas para que el ordenador acepte COGER, TOMAR y LLEVAR, el aventurero se ahorrará mucho de gasto de tiempo innecesario intentando descubrir cuál de estas palabras tiene que usar. Puedes añadir con facilidad tus propias palabras en las líneas de DATA cambiando el bucle FOR ... NEXT de la línea 130 y poniendo los nuevos DATA al final de la línea 145. Tienes que hacer más modificaciones en otras partes del programa, pero ya te diremos más adelante lo que tienes que hacer.

OTRAS RUTINAS

Después de que has completado la lista de verbos con la última rutina, el ordenador necesita algunas rutinas que le permitan atender instrucciones tales como hacer que el aventurero lleve determinados objetos.

La subrutina que comienza en la línea 3010 define V$, N$ e I, que es un número extraído de la matriz R. Con esta corta rutina, la máquina podrá comprender el significado de cada respuesta del aventurero, según el valor de I.

```
500 REM ** ENCONTRAR OPCION
505 IF I=0 THEN GOTO 520
510 ON I GOTO 1010,1150,1240,
    1310,1410,1460,1500,1360,
    1080,1550,3110
520 PRINT:PRINT"NO CONOZCO...
    ";V$:GOTO 370
```

Cada uno de los números que figuran después de la sentencia ON ... GOTO en la línea 510 es el principio de una subrutina. Cada valor de I corresponde a un verbo o grupo de verbos diferente. Por ejemplo, si I = 10, se seleccionará la rutina «matar», es el décimo número de la línea, por lo que la rutina comienza en la línea 1550.

Si la subrutina de comprobación de instrucciones, que comienza en la línea 3010, no encuentra coincidencias para la parte V$ de R$, se asigna a I el valor 0. En tal caso no tendrá efecto la sentencia ON ... GOTO de la línea 510 y se presentará el mensaje de la línea 520.

LA TOMA DE OBJETOS

Ya tienes la rutina para el I = 1, correspondiente al caso en que el aventurero introduce una palabra de dirección; dicha rutina se encuentra en las líneas 1010 a 1060.

Cuando I = 2, significa que el aventurero ha tecleado una palabra de «coger», como COGER, TOMAR o LLEVAR. La siguiente rutina permitirá al aventurero llevarse consigo cualquier objeto que haya en el lugar en que se encuentra. Sería algo así:

```
1140 REM ** RUTINA COJO
1150 FOR G=1 TO NB
1160 IF N$=LEFT$(OB$(G),
     LEN(N$)) THEN 1190
1170 NEXT
1180 PRINT:PRINT"NO COMPRENDO
     ...";N$:GOTO 330
1190 IF OB(G)=-1 THEN PRINT:
     PRINT"YA LO TIENES":
     GOTO 330
1200 IF OB(G)<>L THEN PRINT:
     PRINT"NO ESTA AQUI":
     GOTO 330
1210 PRINT"OK":OB(G)=-1
1220 GOTO 330
```

En las líneas 1150 a 1170 se busca la matriz OB$ que contiene las descripciones cortas de objetos, para saber qué objeto es el designado por el aventurero. Si se encuentra el nombre del objeto, el programa salta a la línea 1190. Si el objeto no figura por ninguna parte de la aventura, la línea 1180 presenta el mensaje NO ENTIENDO, seguido del nombre de objeto tecleado por el aventurero.

Suponiendo que el objeto designado haya sido encontrado, hay que comprobar dos cosas. La línea 1190 examina el elemento de la matriz OB correspondiente a dicho objeto, para ver si ya está en poder del jugador. Si el jugador ya lo tiene (el correspondiente valor de la matriz es −1) se presentará el mensaje YA LO HAS COGIDO.

En la línea 1200 se comprueba si el objeto está presente, examinando nuevamente la matriz de lugares. Si no está presente, el programa dice: NO ESTA AQUI. Naturalmente, puedes cambiar estos mensajes por otros, si no se adaptan bien a tu aventura.

Si el objeto está en el mismo lugar que el aventurero y no ha sido cogido por éste, la línea 1210 dice OK y en el correspondiente elemento de la matriz de lugares se pone el valor −1.

ABANDONO DE OBJETOS

La rutina de «abandono» hace exactamente lo contrario que la anterior. Permite al aventurero dejar los objetos que no quiere llevar con él.

```
1230 REM ** RUTINA DEJO
1240 FOR G=1 TO NB
1250 IF N$=LEFT$(OB$(G),
     LEN(N$)) THEN 1270
1260 NEXT:PRINT:PRINT"NO
     COMPRENDO...";N$:GOTO
     330
1270 IF OB(G)<>-1 THEN PRINT:
     PRINT"NO LO LLEVAS":
     GOTO 330
1280 PRINT"OK":OB(G)=L
1290 GOTO 330
```

Esta rutina funciona de una forma muy parecida a la de «toma». Nuevamente se examina la matriz de descripciones cortas, esta vez se hace en las líneas 1240 a 1260. Si el objeto designado por el aventurero figura en la matriz, la línea 1270 comprueba si el aventurero lo lleva o no. Si no lo lleva, se presenta el mensaje NO LO LLEVAS.

Si el aventurero lleva el objeto, la línea 1280 envía el mensaje OK y se ajusta el correspondiente elemento en la matriz de situación de objetos OB.



P y R

¿Se puede utilizar un sintetizador de voz con una aventura?

Tu aventura podría resultar más interesante programando la máquina de modo que anuncie acústicamente los mensajes, direcciones y descripciones de objetos, en lugar de representarlos en la pantalla.

Consulta el manual de tu sintetizador para ver cómo se puede hacer que la máquina hable y sustituya a las instrucciones que contienen las sentencias PRINT.

Ahora tiene el mismo valor que el lugar actual, es decir L, en vez de −1 que significaba que lo llevaba el aventurero.

LA LISTA DEL BOTIN

Los aventureros desmemoriados estarán muy contentos de poder contar siempre que quieran con una lista de todos los objetos que llevan. Aquí tienes una rutina que hará exactamente eso:

```
1070 REM ** LISTA
1080 PRINT:PRINT"TIENES: ";
     :IN=0
1090 FOR G=1 TO NB
1100 IF OB(G)=-1 THEN PRINT
     TAB(10)OB$(G):IN=IN+1
1110 NEXT
1120 IF IN=0 THEN PRINT
     TAB(10) "NADA"
1130 GOTO 330
```

La línea 1080 envía el mensaje LLEVAS, seguido de la lista de objetos. El bucle FOR...NEXT comprueba todos los elementos de la matriz de situación de objetos. Esta vez los elementos importantes son los que tienen el valor −1, significando que el correspondiente objeto es uno de los que lleva el aventurero. Si el valor de un elemento es −1, se presenta la descripción corta del objeto en cuestión, tomándola de la matriz. El contador de inventario IN se incrementa en 1.

Si el aventurero no lleva ningún objeto, IN se queda a cero y la línea 1120 en vez de la lista de objetos presenta el mensaje NADA.

Las rutinas de «toma», «abandono» e «inventario» pueden utilizarse tal como están , ya que NB ha sido definido en una rutina anterior.

Almacena ahora el programa (SAVE), dejándolo dispuesto para recibir las rutinas finales que veremos la próxima vez. Se trata de las rutinas correspondientes al inspector de hacienda, el ladrillo, la lámpara, el encuentro de la joya, el final de la aventura y, finalmente, la instrucción que describe el objeto de la búsqueda.

Si ejecutas el programa ahora, verás que hay partes del mismo que funcionan mientras que en otras partes ocurren cosas extrañas. La razón es que todavía hacen falta unas cuantas rutinas y el programa salta a líneas que aún no existen.

ELIMINANDO FALLOS

- Asegúrate de que los tres grupos DATA que contienen datos relacionados con los objetos se leen en la matriz adecuada. Si intentas meter datos de cadenas de caracteres en una matriz numérica, recibirás un mensaje de error, o puede que te encuentres con una descripción corta cuando esperabas una larga.

- Ten mucho cuidado en respetar el orden de los DATA en correspondencia con el orden de lectura de las matrices. El orden correcto es: lugar, nombre o descripción corta, descripción larga.

- Después de introducir los objetos haz una pasada de prueba del programa, para asegurarte de que los objetos aparecen en el sitio correcto.

- Al comprobar los objetos, hazlo sobre la retícula, para asegurarte de que no se te ha olvidado nada.

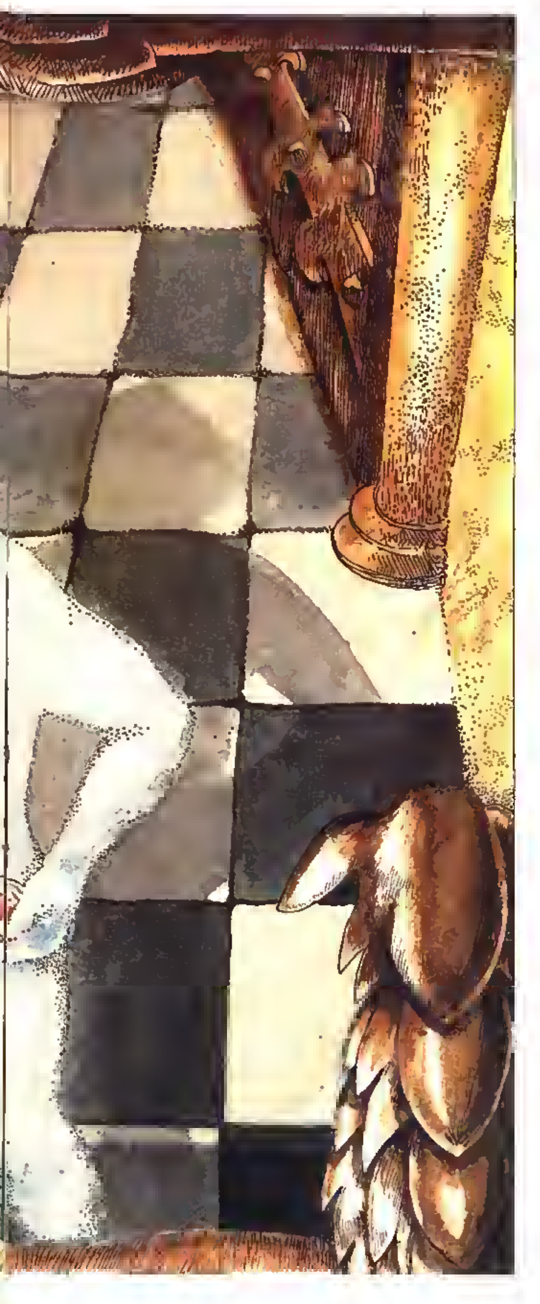


COMPLETANDO LA AVENTURA

- UNA RUTINA OE AYUDA
- EL INSPECTOR OE HACIENOA
- PROBLEMAS CON EL LAORILLO
- ENCENOIENOO LA LAMPARA
- INSTRUCCIONES

La aventura de INPUT está ya easi completa. Sólo quedan las rutinas que la eonvierten en una aventura úniea, las rutinas especiales que confieren a este juego su carácter.

Ya has completado casi todos los elementos que integran tu juego; es el momento de incorporar los últimos detalles del programa. Tienes que añadir aún algunas cosas, como los peligros, las advertencias, y tienes que prever un camino de salida para el caso de que la aventura se complete con éxito. Por último, tienes que poner las instrucciones para jugar.



Dado que muchas de estas rutinas están escritas de forma que contienen detalles específicos de una aventura particular, en los programas que siguen completaremos el juego y te mostraremos en términos generales todo lo que se relaciona con esta fase del programa. En otras aventuras no podrás utilizar estas rutinas sin modificarlas. En el próximo capítulo veremos la forma en que puedes adaptar estos principios para aplicarlos a tus ideas originales.

NECESITAS AYUDA

Si el juego de aventuras que has escrito es bueno, lo más probable es que el aventurero necesite alguna ayuda. Deberás pues prever unas cuantas sugerencias útiles.

Estas sugerencias tendrán la forma de mensajes que el ordenador enviará mediante sentencias PRINT en respuesta a las solicitudes de AYUDA hechas por el jugador. Lo que digan los mensajes y los puntos en que puedan estar disponibles, queda a tu entera discreción, ya que tú eres quien programa la aventura. Si quieres, puedes hacer que no haya un solo mensaje, o hacer que sean deliberadamente engañosos, o bien proporcionar ayuda únicamente en unas cuantas situaciones aisladas. El primer paso para decidir lo que incluir es volver a considerar tu plan original sobre las líneas maestras de la aventura.

En la aventura, hay varios puntos en los que convendría enviar al jugador un corto mensaje. Por ejemplo, podrías advertirle acerca de la habitación oscura, de forma que cuando esté en un lugar contiguo a dicha habitación, la respuesta ante su posible solicitud de ayuda podría ser un mensaje como: MIRA ANTES DE DAR EL SALTO, o incluso algo más críptico.

Otro sitio donde se podría incluir un mensaje de aviso es la orilla del río, donde, en el caso de que se decida a nadar, el aventurero corre el riesgo de ahogarse dependiendo de que lleve o no el ladrillo.

Naturalmente, puedes repasar una a una todas las situaciones en las que podría ser conveniente un poco de ayuda, pero supongamos que decides no ayudar demasiado y enviar un mensaje en un solo punto, el río. Tienes que hacer referencia a este número de lugar, el número 7, y a la variable que registra la presencia del ladrillo, OB(2):

```
3100 REM ** RUTINA AYUDA
3110 IF L<>7 OR OB(2)<>-1
     THEN PRINT:PRINT"LO
     SIENTO, NO TE PUEDO
     AYUDAR AQUI!":GOTO 330
3120 PRINT:PRINT"LOS
     LADRILLOS PESAN MUCHO Y
     HACEN QUE TE DUELA EL
     BRAZO":GOTO 330
```

Si el aventurero no está en la orilla (L<>7), o no lleva consigo el ladrillo (OB(2)<> −1), la línea 3110 presentará el mensaje LO SIENTO, NO TE PUEDO AYUDAR AQUI. Si al llegar a la orilla del río, el aventurero lleva el ladrillo y pide ayuda, la línea 3120 imprimirá el siguienten mensaje de advertencia: LOS LADRILLOS PESAN MUCHO Y HACEN QUE TE DUELA EL BRAZO.

Caso de que quieras hacerlo, no hay nada que te impida incluir una lista

completa de condiciones con sus correspondientes mensajes de aviso.

EL INSPECTOR DE HACIENDA

En la aventura interviene un sujeto que está por ahí merodeando, que es un inspector de hacienda el cual intenta recuperar parte de los impuestos impagados por el aventurero, confiscándole uno de los objetos que lleva. No se preocupa mucho de cuáles son esos objetos, por lo que en algunos casos incluso podría aceptar como pago un ladrillo.

Si el aventurero no tiene la suerte de llevar nada consigo en el momento en que se encuentra con el inspector de hacienda, será encerrado en una mazmorra en la que se pudrirá para siempre. Y aquí termina el juego.

El papel del inspector de hacienda es proporcionar al juego un elemento probabilístico, que resulte impredecible independientemente de cómo se encuentren las demás condiciones. Como en otros ejemplos de introducción de probabilidades en la programación, puedes hacerlo recurriendo a la función RND. Por lo demás, puedes tratarlo como otro objeto cualquiera; la única diferencia es que su situación no es fija, sino que se establece de forma aleatoria.

Aquí tienes las líneas suplementarias que añadir para que aparezca el inspector fiscal:

```
320 R=RND(-TIME):IF INT
    (RND(1)*15+1)=1 AND TA=0
    THEN OB(7)=L:TA=1
480 IF OB(7)=L AND I<>10
    THEN 1590
```

La línea 320 hace que el aventurero tenga una probabilidad sobre 15 de encontrarse con el inspector. Sólo se le permite aparecer una vez durante el juego, por lo que necesitas una variable, TA, para indicar si ha aparecido o no.

Si el número aleatorio es 1 o menor que 1/15, y el inspector todavía no ha aparecido, la línea 320 ajusta el valor que corresponde al inspector de hacienda en la matriz de situación de objetos. La presentación del mensaje del inspector se hace igual que si se tratase de un objeto, almacenándose en la descripción larga de la matriz.

La línea 480 tiene que ver con la supresión del inspector. Se limita a comprobar si has intentado matarle. Si no lo has intentado, el programa salta a la línea 1590.

LA RESPUESTA ANTE EL IMPUESTO

Cuando el inspector vuelve su fea cabeza, sólo hay una solución posible. El aventurero debe disparar contra él utilizando la pistola que se encontró en el río:

```
1540 REM ** RUTINA DISPARO
1550 IF OB(4)<>-1 THEN PRINT"
     CON QUE?":GOTO 320
1560 IF OB(7)<>L THEN PRINT
     V$;" A QUIEN?":GOTO 320
1570 PRINT:PRINT"MATASTE AL "
     ;OB$(7):OB(7)=0:
     GOTO 330
```

Esta rutina se utiliza cuando el aventurero escribe las palabras MATO o DISPARO. Si no lleva la pistola (OB(4) <> −1), la línea 1550 presentará el mensaje CON QUE? Análogamente si el inspector de hacienda no está presente y el jugador intenta matarle, la línea 1560 le pregunta A QUIEN?

La línea 1570 le dice al aventurero MATASTE AL INSPECTOR DE IMPUESTOS, y ajusta la matriz de situaciones de los objetos de forma que dicho inspector ya no existe.

LA VENGANZA DEL INSPECTOR

El aventurero se encuentra con lo siguiente:

```
1580 REM ** INSPECTOR FISCAL
1590 IN=0:OB(7)=0
1600 FOR K=1 TO NB
1610 IF OB(K)=-1 THEN
     IN=IN+1
1620 NEXT
1630 IF IN<>0 THEN 1640
1635 PRINT:PRINT"COMO NO
     LLEVAS NADA TE
     ENCIERRA "
1638 PRINT"EN UNA FRIA
     MAZMORRA":GOTO 1360
1640 R=RND(-TIME):K=INT
     (RND(1)*NB+1):IF OB(K)
     <>-1 THEN 1640
1650 PRINT "EL INSPECTOR TE
     CONFISCA EL OBJETO ";OB$
     (K):OB(K)=0:GOTO 330
```

Al inspector sólo se le permite aparecer una vez durante toda la aventura, por lo que la línea 1590 ajusta la matriz de situación de objetos poniendo a cero el que corresponde al inspector; OB(7). Esto no tiene efecto alguno en esta rutina, pero hace que una vez que salgamos de ella, el inspector no vuelva a aparecer. IN es un contador utilizado para comprobar si se llevan objetos.

En las líneas 1600 a 1620 se recorre toda la matriz de situaciones de objetos, para comprobar si cada uno de los objetos se lleva o no. Por cada objeto que se porte, se incrementa IN en uno.

Si el aventurero no lleva objeto alguno, el valor de IN permanece a cero y se presenta el siguiente mensaje: COMO NO LLEVAS NADA, TE ENCIERRA EN UNA FRIA MAZMORRA. El juego termina aquí, y se pregunta de nuevo al aventurero si quiere jugar otra vez saltando a la línea 1360.

Si por el contrario el aventurero lleva objetos consigo, en la línea 1640 se elige uno al azar. Si el número elegido corresponde a uno de los objetos transportados por el aventurero, dicho objeto es confiscado; si por el contrario dicho objeto no ha sido cogido, se selecciona otro número al azar, siguiendo así hasta que el número elegido corresponda a uno de los objetos transportados.

Después de que ha sido seleccionado un objeto, la línea 1650 informa al aventurero de que dicho objeto ha sido confiscado por el inspector. La matriz de situación de objetos queda modificada de forma que dicho objeto ya no existe.

TOMANDO UN BAÑO

La siguiente rutina se utiliza cuando el aventurero decide cruzar el río a nado:

```
1400 REM ** RUTINA NADO
1410 IF L<>7 THEN PRINT
     "EN QUE?!!":GOTO 330
1420 IF OB(2)=-1 THEN PRINT
     "QUE DESASTRE, TE HAS
     HUNDIDO":GOTO 1360
1430 IF OB(4)>-1 THEN PRINT
     "HAS ENCONTRADO UNA
     PISTOLA":OB(4)=-1:GOTO 3
     30
1440 PRINT"ESTAS TODO MOJADO"
     :GOTO 330
```

En la línea 1410 se comprueba si el aventurero está en el río. De no ser así, presenta la pregunta DONDE? Como en esta aventura no hay piscinas ni océanos, no hay que preocuparse de poner una rutina que se ocupe de las posibles respuestas a esta pregunta. No aparece ningún mensaje más y el juego continúa.

Si el aventurero intenta lanzarse al río a nadar cargado con el ladrillo, se muere: QUE DESASTRE, TE HAS HUNDIDO. Después de hundirse, puede reencarnarse cuando se le pregunta si quiere intentarlo otra vez.

En la línea 1430 se comprueba si el aventurero lleva la pistola, de no ser así se modifica la matriz de situación de objetos y aparece el mensaje HAS ENCONTRADO UNA PISTOLA.

Si el aventurero ya ha encontrado la pistola y por cualquier razón intenta cruzar de nuevo el río a nado, la línea 1440 le dice ESTAS MOJADO.

AL FIN, LA JOYA

El aventurero sólo puede hallar la fabulosa joya en la bolsa de canicas que ha encontrado. El paso necesario para ello es que vacíe dicha bolsa, con lo que aparecerá la joya. He aquí la correspondiente rutina:

```
1450 REM ** RUTINA VACIO
1460 IN=0:IF N$=LEFT$("BOLSA'
     ,LEN(N$)) THEN IN=1
1465 IF IN<>1 THEN PRINT
     "NO PUEDES VACIARLO":
     GOTO 330
1470 IF OB(1)<>-1 THEN G=1
     :GOTO 1270
1480 PRINT "LAS CANICAS
     RUEDAN POR EL SUELO"
     :OB(5)=L:GOTO 370
```

Esta rutina es llamada cada vez que el aventurero ordena VACIAR algo. En la línea 1460 se comprueba si ese algo es una bolsa. Si no es así, (N$ <> «BOLSA»), aparece el mensaje NO PUEDES VACIARLO. La línea 1470 es para comprobar si la bolsa está en-



tre las pertenencias del jugador (OB(1)<> −1). De no ser así, en vez de presentar un nuevo mensaje, el programa salta a la línea 1270 para aprovechar el mensaje NO LO TIENES, que ya ha sido incluído en el programa.

Si la bolsa está presente, el programa llega a la línea 1480. Aparece el mensaje: LAS CANICAS RUEDAN POR EL SUELO, y se ajusta la matriz de situación de objetos de forma que la joya está ahora en la situación actual.

No hace falta presentar mensaje alguno para esto, ya que al saltar a la línea 370 se puede utilizar el mecanismo habitual de descripción larga. En la pantalla aparece pues la descripción que pusiste en la matriz de descripción larga (línea 240).

ENCENDIENDO LA LAMPARA

Cuando el aventurero quiere ver las salidas de que dispone para abandonar la habitación oscura, necesita encender la lámpara. Si no lleva consigo la lámpara, no tendrá forma de disipar la oscuridad y se quedará allí atascado. Aquí tienes la rutina de encendido de la lámpara:

```
1490 REM ** RUTINA LUZ
1500 IN=0:IF N$=LEFT$
     ("LAMPARA",LEN(N$))
     THEN IN=1
1505 IF IN<>1 THEN PRINT
     "NO PUEDES HACERLO":
     GOTO 330
1510 IF OB(6)<>-1 THEN G=6
     :GOTO 1270
```



```
1520 IF LA=1 THEN PRINT
     "YA ESTA ENCENDIDA"
     :GOTO 330
1530 LA=1:PRINT"OK":GOTO 330
```

Cada vez que el aventurero ordene LUZ, se llamará a esta rutina. La línea 1505 es muy parecida a la correspondiente línea de la rutina de «vaciado», comprobando si el aventurero ha mencionado la lámpara. El mensaje NO PUEDES HACERLO, aparece exactamente de la misma forma que antes.

La línea 1520 comprueba si el indicador de «lámpara encendida», LA, ha sido activado, y le dice al aventurero si ya está encendida la lámpara.

El indicador de lámpara encendida se pone a 1 en la línea 1530, que también presenta el mensaje O.K.

EL FINAL ESTA CERCA

En el salón del trono está colgando la cadena y el aventurero acaba de entrar en escena.

¿Qué tiene que hacer? ¿Qué pasa si se tira de la cadena? Aquí tienes una rutina en la que se contemplan las consecuencias:

```
1300 REM ** RUTINA TIRO
1310 IN=0:IF N$=LEFT$
     ("CADENA",LEN(N$)) THEN
      IN=1
1315 IF IN=1 AND L<>24 THEN
     PRINT "NO SUCEDE NADA":
     GOTO 330
1320 IF IN<>1 THEN PRINT "NO
     PUEDES TIRAR DE ESO!"
     :GOTO 330
1330 IF OB(5)=-1 THEN 1340
1335 PRINT "AL TIRAR DE LA
     CADENA HAS SIDO":PRINT
     "ARRASTRADO POR EL AGUA
     Y"
1338 PRINT "DESAPARECES POR
     EL EXCUSADO, CANERIA"
     :PRINT"ABAJO":GOTO 1360
1340 REM ** FIN DE LA
     AVENTURA
1350 PRINT "BIEN HECHO,
     ACABAS DE COMPLETAR LA":
     PRINT"AVENTURA"
1360 PRINT:PRINT"OTRO JUEGO
     (S/N)?"
1370 A$=INKEY$:IF A$<>"S"
     AND A$<>"N" THEN 1370
1380 IF A$="S" THEN RUN
1390 PRINT:PRINT:END
```

La línea 1310 contempla la posibilidad de que el aventurero haya cogido la cadena fuera del salón del trono antes de tirar de ella. Dirá al aventurero: NO SUCEDE NADA.

Si el aventurero intenta tirar de cualquier otro objeto de la aventura, recibe el mensaje: NO PUEDES TIRAR DE ESO, contenido en la línea 1320.

Después ocurre lo inimaginable. Si el aventurero se encuentra en el salón del trono, pero no ha encontrado la joya, se le envía el mensaje: AL TIRAR DE LA CADENA, HAS SIDO ARRASTRADO POR EL AGUA Y DESAPARECES POR EL EXCUSADO CAÑERIA ABAJO. Y de esta forma termina el juego.

Si por el contrario el jugador de la aventura sí ha encontrado la joya y tira de la cadena en el salón del trono, ninguna de las líneas anteriores tendrá efecto y podrá exhalar un suspiro de alivio cuando lea: BIEN HECHO, ACABAS DE COMPLETAR LA AVENTURA.

Por último, en las líneas 1360 a 1380 se presenta una opción para jugar otra vez. Realmente sólo resulta útil en caso de que el aventurero haya quedado atrapado en la mazmorra o haya sido engullido por el inodoro.

LAS INSTRUCCIONES

En este momento ya dispones de un juego de aventuras que funciona a la perfección, por lo que ha llegado el momento de darle los últimos toques.

Si no se le dan instrucciones, el aventurero no podrá saber el objetivo de todos tus esfuerzos, ni lo que tiene que hacer. Antes de añadirle a un juego el conjunto de instrucciones, comprueba la cantidad de memoria que te queda disponible. Si queda poca, es hora de eliminar todas las sentencias REM, aunque puede ser que ello te obligue, para evitar errores, a cambiar la numeración de los GOSUB que envían el programa hacia ellas.

La cantidad de instrucciones a incluir es algo que conviene considerar con cuidado. Tienes que tomar una decisión dependiente de la cantidad de memoria disponible, de cuántas sugerencias quieras dar en cada etapa, y de otras consideraciones tales como el formato de la pantalla de tu máquina, que afectará en gran medida al grado de detalle que puedas dar antes de tener que pasar a otra pantalla.

Como la aventura de **INPUT** es muy sencilla, la rutina de instrucciones es corta y contiene poca informacion. Aquí la tienes:

```
10 PRINT"QUIERES VER LAS
   INSTRUCCIONES?"
20 A$=INKEY$:IF A$=""
   THEN 20
30 IF A$="S" THEN GOSUB 6000
6000 REM ** INSTRUCCIONES
6010 PRINT:PRINT"A CAUSA DE
     UNA CRISIS ECONOMICA HAS
     ":PRINT"HUIDO DE TU PAIS
     "
6020 PRINT:PRINT"LA SOLUCION
     A TUS PROBLEMAS ESTA
     EN"
6025 PRINT"ENCONTRAR EL GLOBO
      OCULAR,"
6027 PRINT"PASAR AL FINAL Y
     SUPERAR"
6029 PRINT"LA PRUEBA DE
     INICIATIVA"
6030 PRINT:PRINT"EVITA A TODA
     COSTA AL INSPECTOR
     FISCAL"
6040 PRINT:PRINT"PULSA UNA
     TECLA"
6050 A$=INKEY$:IF A$=""
     THEN 6050
6060 RETURN
```

Ahora ya puedes almacenar (SAVE) en cinta la aventura completa.

En el próximo capítulo, veremos la forma de utilizar la estructura que hemos seguido a lo largo del juego del Ojo Precioso de la Imagen Pùrpura, para que sirva de base a tus propias aventuras.

AVENTURAS: EL SIGUIENTE PASO

Las aventuras se parecen al tabaco: producen hábito; además las puedes comprar ya empaquetadas o *liar* las tuyas propias. Aquí verás la manera de usar la aventura como base para desarrollar la tuya propia.

En este momento tienes almacenado en cinta un juego completo de aventuras que funciona perfectamente. Al ir recorriendo todo su desarrollo, has visto cómo se van combinando todos los elementos que la constituyen, partiendo de un bosquejo muy rudimentario de la historia. En este capítulo veremos la manera de utilizar dicho juego como punto de partida para el desarrollo de tus aventuras domésticas.

Aunque en los anteriores capítulos hemos visto algunas sugerencias sobre posibles variaciones del juego, seguidamente nos ocuparemos con más profundidad de la forma en que lo puedes hacer.

No siempre será posible ser muy específico acerca de las alteraciones a introducir, ya que muchas de ellas dependerán totalmente de la aventura que estés escribiendo, pero muchas de ellas serán muy fáciles de incorporar siguiendo las instrucciones que veremos más adelante. Al principio puede que algunas de las técnicas te desanimen un poco, pero si empiezas tratando de escribir una aventura corta y sencilla, en seguida captarás los principios básicos. Durante las primeras fases no pretendas hacer demasiadas alteraciones al mismo tiempo; es mejor que te limites a ir recorriendo sistemáticamente las secciones de este artículo y no tendrás muchos problemas.

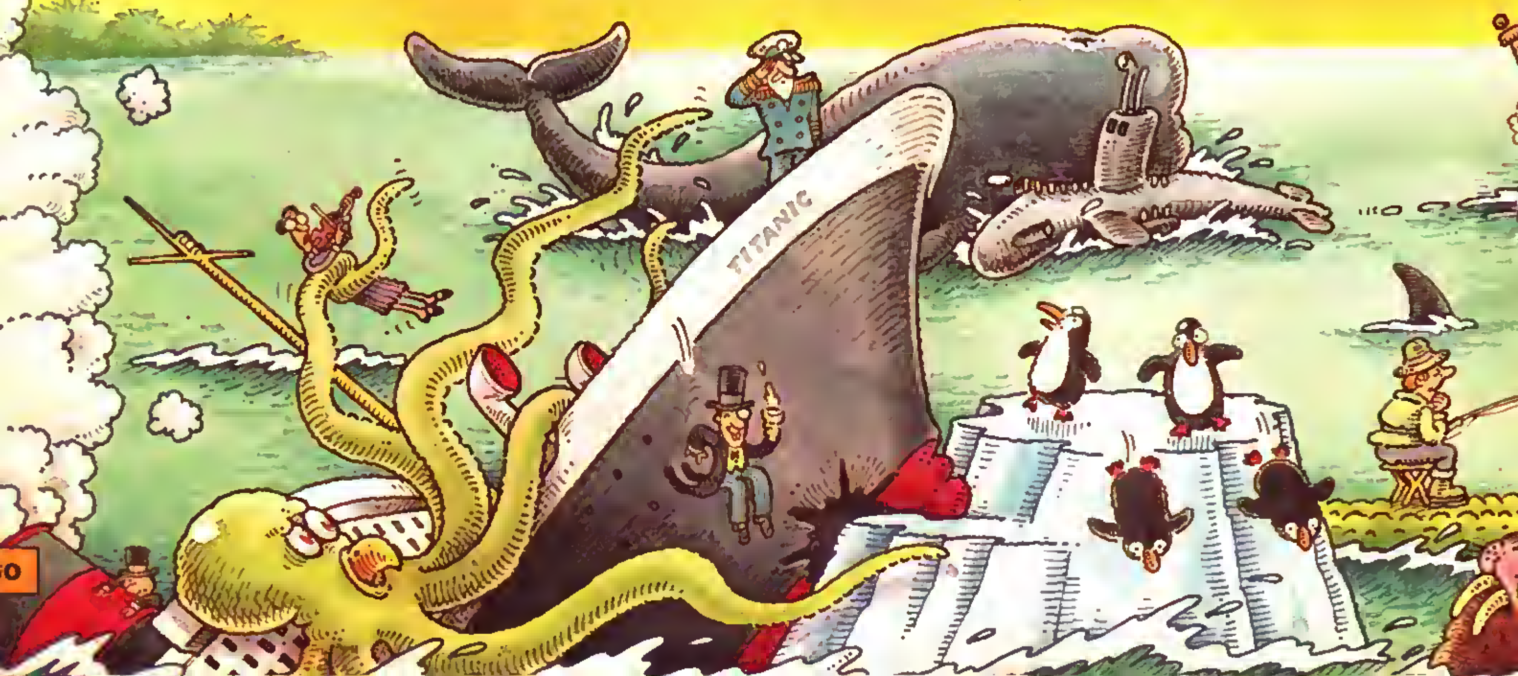
TEMAS PARA TU PROPIA AVENTURA

Antes de ponerte a escribir un juego de aventuras propio, tienes que inventarte un bosquejo de la historia que te sirva como punto válido de partida.

La estructura argumental de las aventuras de mayor éxito suele ser bastante tradicional: hay un principio, una fase intermedia, y un final, con una secuencia impuesta por el orden en que se quiere que aparezcan los enigmas que haya que resolver. Sin embargo es una suerte que no seas una **Agatha Christie** a la hora de escribir juegos de aventuras, ya que aunque hay montañas de ideas posibles, su realización no resulta fácil en las primeras etapas. A continuación presentamos unas cuantas sugerencias que harán más fácil tu labor.

Podrías estructurar la aventura en torno a una novela policíaca. El punto de partida podría ser una habitación en la que hay un cadáver con un puñal clavado, y el objetivo final del juego sería encontrar quién es el asesino. Tal vez prefieras utilizar el personaje de un mayordomo en lugar del inspector de hacienda. Su papel podría consistir en ayudar al aventurero, o en estorbarle.

Hay varias maneras de utilizar un argumento de naufragios en la aventura. Puedes hilvanar una historia tradicional de náufragos y piratas, o hacer que tu aventurero sea el único superviviente de un accidente aéreo. Si prefieres situar tu aventura en el futuro, puedes montar un desastre espacial que haga que el aventurero se encuentre abandonado en un



- PENSANDO UNA NUEVA AVENTURA
- POSIBLES TEMAS
- EXTENSION DE LA RETICULA
- NUEVOS OBJETOS



planeta hostil a años luz de su civilización y con una nave espacial averiada. El objeto de la aventura sería en este caso encontrar alguna vía de escape. Los lugares pueden muy bien ser hostiles, y hay muchas posibilidades de encontrar rutas de escape imaginativas (y escondidas).

Otras aventuras de este tipo pueden consistir en escaparse de **Alcatraz**, o **Sing Sing**, o la **Isla del Diablo**, el nombre que más te guste. Puedes encontrar fuentes de inspiración en cualquiera de los múltiples libros sobre evasiones famosas que han sido publicados, y si puedes hacerte con un plano real del lugar, resultará mucho mejor el planteamiento de los lugares de tu aventura.

Otro de los temas tradicionales, que constituirá las delicias de muchos diseñadores de aventuras, son los espías; también se incluye aquí el espionaje industrial, por ejemplo el robo de un diseño de ordenador de la competencia.

También puedes sacar multitud de temas de la Historia. Por ejemplo las Cruzadas son una evidente fuente de inspiración para los juegos de aventuras; lo mismo puede decirse de cualquier campaña militar.

Por último, un tema que se ha convertido en tópico: una aventura basada en un holocausto nuclear. Las posibilidades son enormes: mutantes, búsqueda de los trajes antirradiación, grupos de bandoleros que merodean muriéndose de hambre, intentando encontrar comida y agua sin contaminar, etc. etc,etc.

¿MAS LUGARES?

La aventura de **INPUT** es mucho más corta de lo que suele ser la longitud normal de estos juegos, por lo que pronto te encontrarás que tus propios juegos de aventuras superan ampliamente a este programa.

Sigue las instrucciones de las páginas 38 a 43 para obtener una retícula adecuada a tu programación. La retícula de **INPUT** tiene 6×4 lugares, en total 24, de los que sólo se utilizan 12. Si decides trabajar adaptándote a esta retícula, puedes hacer una de dos cosas: o bien modificas el programa existente, lo que representa menos trabajo aunque es más difícil de descifrar, o tecleas un programa completamente nuevo, lo cual representa algo más de esfuerzo, aunque puede que te resulte menos confuso. Dependiendo de la elección que hagas, puedes cargar (LOAD) el programa existente desde la cinta, o, si tienes una impresora, listarlo en papel. Las adaptaciones que siguen a continuación dependen del tamaño de la retícula que te resulte útil. Si tienes 24 lugares o menos, puedes usar la retícula existente tal como está, dibujando el mapa sobre la misma. Si tu mapa requiere una retícula mayor, dibújala y numérala de la forma que ya sabes.

Después de organizar la retícula, puedes empezar a introducir tu propio juego de descripciones de lugares en la máquina. Tienen que sustituir a las descripciones de los lugares existentes a partir de la línea 5000.

Cada descripción de un lugar ha de ir seguida con la línea que contiene las posibles salidas del mismo, tal como ocurría con el programa original. Las variables N, S, E y O corresponden a Norte, Sur, Este y Oeste. Pueden tomar los valores 0 y 1; 0 siginifica que no hay salida en esa dirección, mientras que 1 significa que sí hay una salida. El esfuerzo extra de teclear unas cuantas líneas suplementarias de sentencias REM con los números de los lugares es algo que te va a merecer la pena.

El siguiente paso es modificar las sentencias ON ... GOSUB de las líneas 330 a 350. El primer número que sigue a la sentencia GOSUB de la línea 330 es el número de línea en que el ordenador encontrará la descripción del lugar 1. Si no hay un lugar 1 en la aventura —no tienes porqué utilizar todas las casillas de la retícula— se introduce en su lugar un cero. El siguiente número corresponde al número de línea del segundo lugar, y así sucesivamente. Tiene que haber un número para cada uno de los lugares de la retícula.

MOVIMIENTO

Si has diseñado un juego basado en una retícula de tamaño diferente al de la utilizada en la aventura, necesitarás modificar las rutinas de movimiento de las líneas 1000 a 1040. Más específicamente, si la retícula no tiene una anchura de seis cuadros, tendrás que cambiar las líneas de Norte y Sur (líneas 1010 y 1030) , ya que para cambiar de fila lo que hacías era sumar o restar seis. Para hacer la modificación, no tienes más que contar de cuántas casillas se compone la fila de tu retícula, y sustituir el número 6 por el valor del nuevo ancho.

LOS OBJETOS

Los objetos de tu nueva aventura serán diferentes de los de la aventura de **INPUT**, por lo que es probable que tengas que hacer cambios bastante extensos en las líneas 160 a 260.

Cuenta el número de objetos que vayas a utilizar en tu nueva aventura. Este número determina el valor de NB y debe ser el primer dato de la línea 200, siendo utilizado para dimensionar las matrices de la línea 180, y para los bucles FOR ... NEXT de otras partes del programa.

Aunque resulta más claro utilizar una línea de programa separada para cada objeto, si has escrito un juego que utilice muchos objetos, puede que te resulte más cómodo poner más de un objeto en cada línea. Cualquiera que sea la forma en que decidas ponerlos, tus datos deben guardar el orden correcto, ya que cada uno de los tres grupos de datos forman parte de matrices diferentes. El orden es el siguiente: número de lugar, descripción corta y descripción larga. Si el objeto no aparece hasta más tarde en la aventura, tal vez debido a que el aventurero lo encuentra, o es de aparición aleatoria, como es el caso del inspector de hacienda, el correspondiente número de lugar será un cero.

NUEVAS PALABRAS

Haz una lista de todas las instrucciones que el ordenador debe esperar recibir del aventurero durante el juego. Dicha lista incluirá palabras sencillas, tales como las órdenes y las palabras AYUDA e INVENTARIO, y órdenes formadas por dos palabras, tales como COGER LAMPARA o MATAR POSADERO.

Las entradas a base de dos palabras se desdoblan en V$ y N$, verbos y nombres, aunque esta denominación no siempre corresponda estrictamente a la definición gramatical.Tu interés debe centrarse en todas las palabras sencillas y en la primera palabra de cada pareja. Para los fines del programa, las primeras palabras son los verbos, V$. Agrupa los diferentes verbos con arreglo a su significado, por ejemplo COMER y MASTICAR, o bien OLER y HUSMEAR. Cada uno de estos grupos necesitará un número que también deberás anotar. No importa cómo se asigne ese número; basta que sepas que cada número se re-

fiere a un determinado grupo de palabras.

Ya puedes modificar el programa. La rutina de manejo de verbos está en las líneas 110 a 150. Los verbos y sus correspondientes números se introducen como datos en las líneas 140 y 150, como pares, cuyo primer componente es el número y el segundo el verbo.

No te olvides de volver a dimensionar las matrices de la línea 120 y de ajustar el bucle FOR ... NEXT de la línea 130 adaptándolo al número total de objetos que quieras usar.

RUTINAS DE VERBOS

Cada una de las categorías separadas de verbos, es decir, cada uno de los números, requerirá una rutina separada.

Es difícil dar instrucciones explícitas acerca de la forma de escribir estas rutinas, debido a que una buena parte de las rutinas de cualquier aventura no valdrá para las demás.

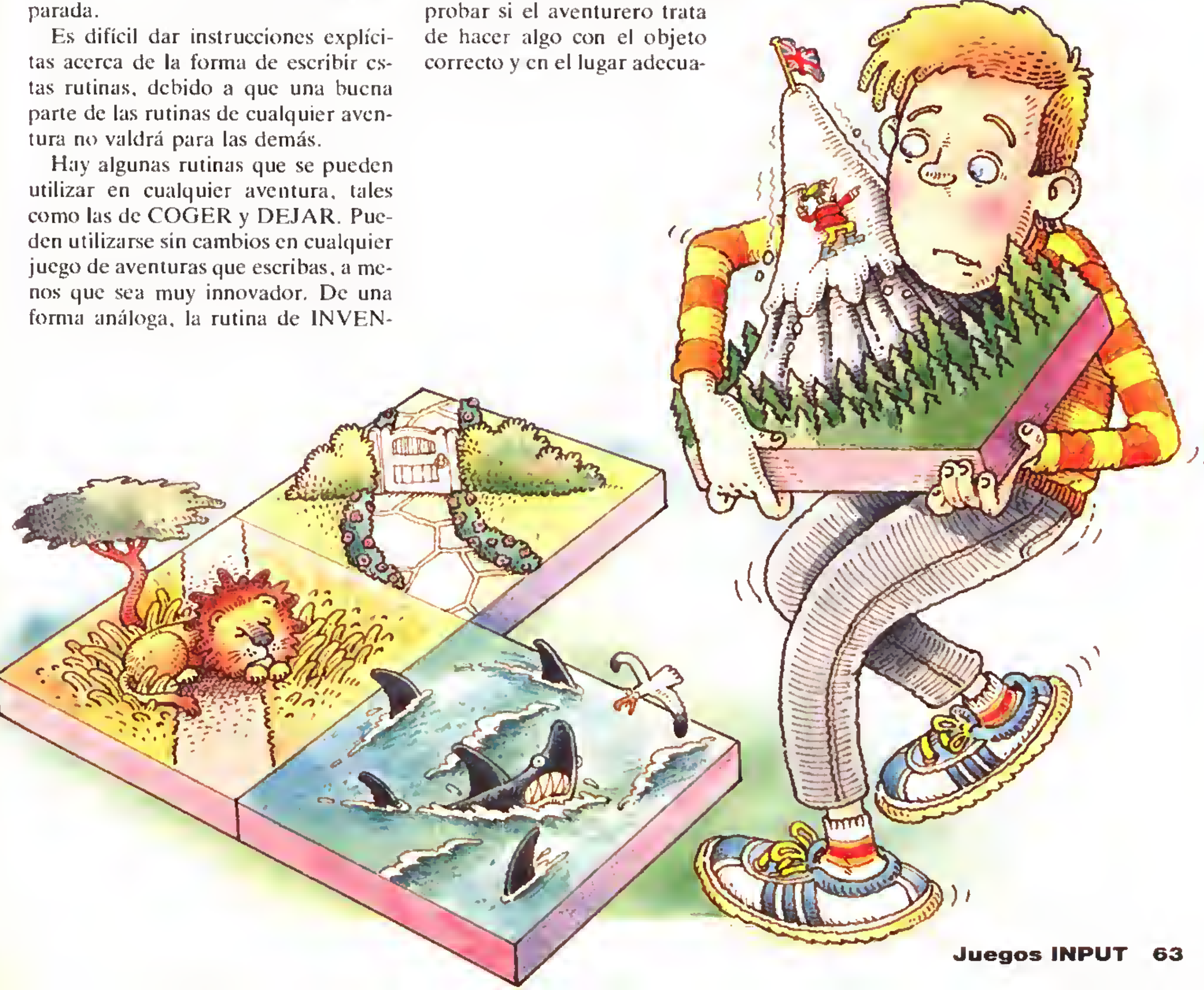
Hay algunas rutinas que se pueden utilizar en cualquier aventura, tales como las de COGER y DEJAR. Pueden utilizarse sin cambios en cualquier juego de aventuras que escribas, a menos que sea muy innovador. De una forma análoga, la rutina de INVENTARIO (líneas 1070 a 1130) es la misma para cualquier aventura, por lo que puedes utilizarla sin cambios en la medida en que la matriz es la misma y que NB —el número de objetos— tiene el mismo significado en la nueva aventura.

Otra rutina, que podría ser de aplicación sería la rutina de encendido de la lámpara, porque el encender y apagar lámparas y linternas es una ocupación muy frecuente en los juegos de aventuras. Dicha rutina está situada en las líneas 1490 a 1530.

Probablemente las demás rutinas no son lo bastante generales como para trasladarlas en bloque, pero hay algunos puntos que conviene que tengas en cuenta cuando escribas tus propias rutinas de verbos. Básicamente las rutinas se ponen ahí para comprobar si el aventurero trata de hacer algo con el objeto correcto y en el lugar adecuado. Si el lugar está equivocado, el programa presentará un mensaje de que eso es apropiado para ciertas situaciones, pero AHI NO. Ocurra lo que ocurra, cerciórate de que el aventurero conoce cuál fue el efecto de su última instrucción, en otras palabras, para cualquier cosa que se le diga a la máquina que haga, debe aparecer en pantalla un mensaje de respuesta.

Cuando tengas lista tu rutina de verbos, introdúcela en el programa. Si numeras el programa de forma análoga a la aventura de **INPUT**, el lugar para esta rutina estará entre las líneas 1070 y 2999.

El ordenador tiene que poder selec-



cionar la rutina correcta de acuerdo con el verbo utilizado por el aventurero. Para que pueda hacer esto tienes que modificar la línea 510.

Lo único que tienes que hacer para ello es observar tu lista de números de verbos. A continuación, utilizando ese orden numérico, pon después de la sentencia ON ... GOTO, las líneas de comienzo de la rutina correspondiente a cada verbo.

RUTINA DE AYUDA

La rutina final a la que debes dedicar tu atención es la de AYUDA. Considera en qué puntos de tu aventura podría ser necesaria una sugerencia, y utiliza una línea IF ... THEN para hacerla.

Hay otros detalles que puede que requieran modificación, dependiendo de las características de tu aventura; tal es el caso de la línea 320, que hace que aparezca el inspector de hacienda. No pierdas de vista tampoco el lugar de comienzo, que se establece en la línea 280.

VARIABLES Y MATRICES

Ahora que ya sabes «meterte dentro» del programa de la aventura, aquí tienes una lista de las variables y matrices junto con el uso a que se destinan:

R$() matriz de verbos y respuestas.

R() matriz de números de respuestas.

Los elementos correspondientes de las dos matrices anteriores son los pares de verbos y los significados.

OB() matriz con el número de lugar para cada objeto.

OB$() matriz de descripciones cortas de los objetos.

S1$() matriz de descripciones largas de los objetos.

Los elementos correspondientes de las tres matrices anteriores contienen información relativa a cada objeto en particular.

NB número de objetos de la aventura. Se utiliza para dimensionar las matrices y en los bucles FOR ... NEXT.

L situación actual del aventurero.

LA indicador de estado de la lámpara. Se pone a 1 cuando está encendida y a 0 cuando está apagada.

TA indicador del inspector de hacienda.

N,S,E,O direcciones de salida. Se ponen a 1 si existe una salida en esa dirección y a 0 si no existe.

I$ entrada total antes de ser desglosada en verbos y nombres.

V$ parte de verbos de I$.

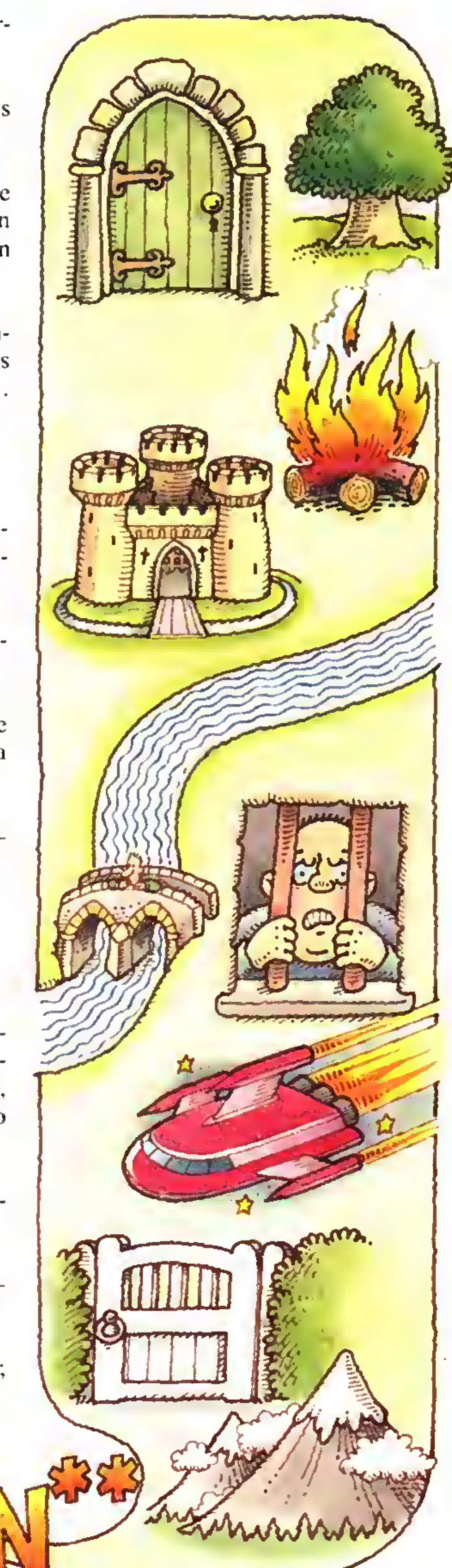
N$ parte de nombres de I$.

I número correspondiente al significado de un determinado verbo. Se utiliza para dirigirse a la rutina correcta, que es la que se ocupa de ese verbo en particular.

IN número de objetos del INVENTARIO.

A$ respuesta a la pregunta QUIERES PROBAR OTRA VEZ?

G número de objeto abandonado; G es un elemento de la matriz OB.



REM**LOCATION**

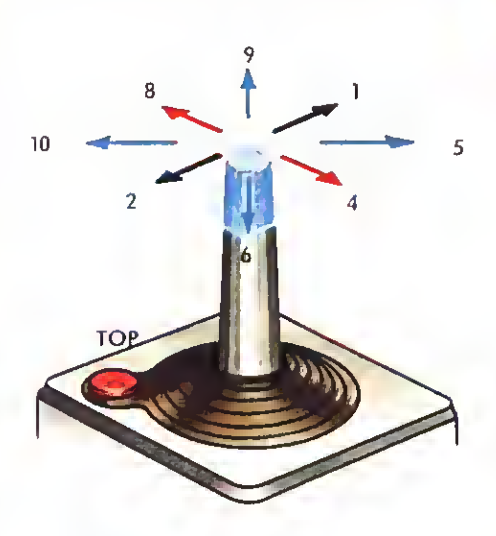
LA PROGRAMACION PARA JOYSTICKS

- COMO LEER LOS JOYSTICKS DESOE EL BASIC
- HAZ TUS JUEGOS MAS PROFESIONALES
- EL PULSAOOR OE FUEGO

Los *joysticks* son la clave para tener unos juegos más profesionales. Pero el hacer que tus juegos resulten más divertidos no significa que tengas que aprender código máquina; puedes empezar con el BASIC.

Una diferencia evidente entre los juegos comerciales y los producidos en casa es con frecuencia la existencia de una opción para *joystick*. Sin embargo, no tienes que sumergirte en las profundidades intrincadas del código máquina para utilizar los *joysticks* en tus propios programas.

En esta sección del coleccionable, veremos la manera de utilizar los *joysticks* con programas escritos en BASIC, haciendo que tus programas de juegos resulten más profesionales y más divertidos de jugar.



En el próximo capítulo utilizaremos la rutina de *joystick* en un juego, por lo que no debes de olvidarte de almacenar el programa.

Lo primero que necesitas es un *joystick* adecuado. Todo programa está escrito para adaptarse a las características de los *joysticks* indicadas.

Existe una gran variedad de *joysticks* disponibles, pero todos están estandarizados para trabajar con vuestros ordenadores MSX.

UN PUNTO DE MIRA

Teclea la siguiente sección de programa y tendrás un punto de mira controlable con tu *joystick*:

```
10 CLS:KEYOFF:COLOR5,1,1:SCREEN0,0,0
120 A=120:B=100
130 A4$=CHR$(&H0)
140 A5$=CHR$(&H10)
150 A6$=CHR$(&H38)
160 A7$=CHR$(&H10)
170 A8$=CHR$(&H0)
250 A$=A4$+A5$+A6$+A7$+A8$
280 SCREEN 2,3,0
290 SPRITE$(0)=A$
470 LINE (0,0)-(256,45),7,BF
480 CIRCLE (150,40),20,11,,,1.4
490 PAINT (150,40),11
500 LINE (0,45)-(256,191),5,BF
510 S=STICK(0)
520 IF S=1 THEN B=B-10
530 IF S=2 THEN A=A+15:B=B-10
540 IF S=3 THEN A=A+15
550 IF S=4 THEN A=A+15:B=B+10
560 IF S=5 THEN B=B+10
570 IF S=6 THEN A=A-15:8=8+10
580 IF S=7 THEN A=A-15
590 IF S=8 THEN A=A-15:B=B-10
600 IF A=>250 THEN A=250
610 IF A=<10 THEN A=10
620 IF B=>181 THEN B=181
630 IF B=<10 THEN B=10
640 PUT SPRITE 0,(A,B),1,0
840 GOTO 510
```

El programa empieza inicializando la pantalla en la línea 10, mientras la línea 120 define la posición del punto de mira que se empleará más tarde. A continuación se dibuja el punto de





mira creando un sprite entre las líneas 130 a 170; en la línea 280 preparamos la pantalla en modo de gráficos de alta resolución y de sprites ampliados; con las líneas 470 a 500 dibujamos un fondo que, aunque no es preciso para este programa, lo será cuando lo unamos con el de la próxima sección.

Una vez definido el sprite debemos utilizar un comando que controle nuestro *joystick*. Este comando es STICK. El número entre paréntesis selecciona el control desde el cursor; si es desde el primer *joystick* es 1; si es desde el segundo, es 2. De la línea 520 en adelante chequeamos todas las posiciones del *joystick*; cuando este chequeo detecta un cambio, el programa cambia los valores A o B encargados de almacenar la posición de la mira. Los valores que puede controlar el comando STICK y que dependen de la posición del *joystick* son: si detecta "1" arriba; "2" arriba a la derecha; "3" derecha; "4" derecha abajo; "5" abajo; "6" abajo a la izquierda; "7" izquierda; "8" izquierda arriba; cuando el *joystick* no esté en movimiento, el valor de STICK será "Ø".

Desde la línea 600 a 630 se controlan las posiciones de la mira para que no salga de la pantalla y aparezca por el lado opuesto. La línea 640 representa el punto de mira en pantalla, y, por fin, en la línea 840 devuelve el control del ordenador a la línea 510 para que chequee de nuevo las posiciones que puede adoptar el *joystick*.

P y R

¿Se le puede agregar una rutina de "joystick" a cualquiera de los juegos que aparecen en INPUT?

Sí, esto no encierra gran dificultad: simplemente hay que variar el núcleo del programa, el cual lee el teclado mediante la sentencia INKEY y sustituirlo por una rutina que lea el valor de STICK como la que hay en este programa y que está situada entre las líneas 52Ø y 54Ø.

EL JUEGO DE LA CAZA DE PATOS

Para los que no puedan esperar a que se levante la veda o no quieran tiritar de frío en el campo, aquí presentamos una rutina de disparo contra los patos, que podréis utilizar junto con la rutina de *joystick*.

Si has seguido atentamente el capítulo anterior, tendrás guardado en cinta un programa para mover por la pantalla el punto de mira. Pero aunque resulte satisfactorio disponer de un programa de este tipo escrito en BASIC, tal como está no vale para mucho.

Por eso el siguiente paso es utilizar la nueva rutina dentro de un programa de juegos. Al añadir las siguientes líneas de programa especialmente escritas para tu máquina, tendrás un juego de caza de patos salvajes, aunque naturalmente puedes dibujar tus propios gráficos y disparar contra aviones, búfalos, dirigibles o cualquier otra cosa que te sugiera tu fantasía.

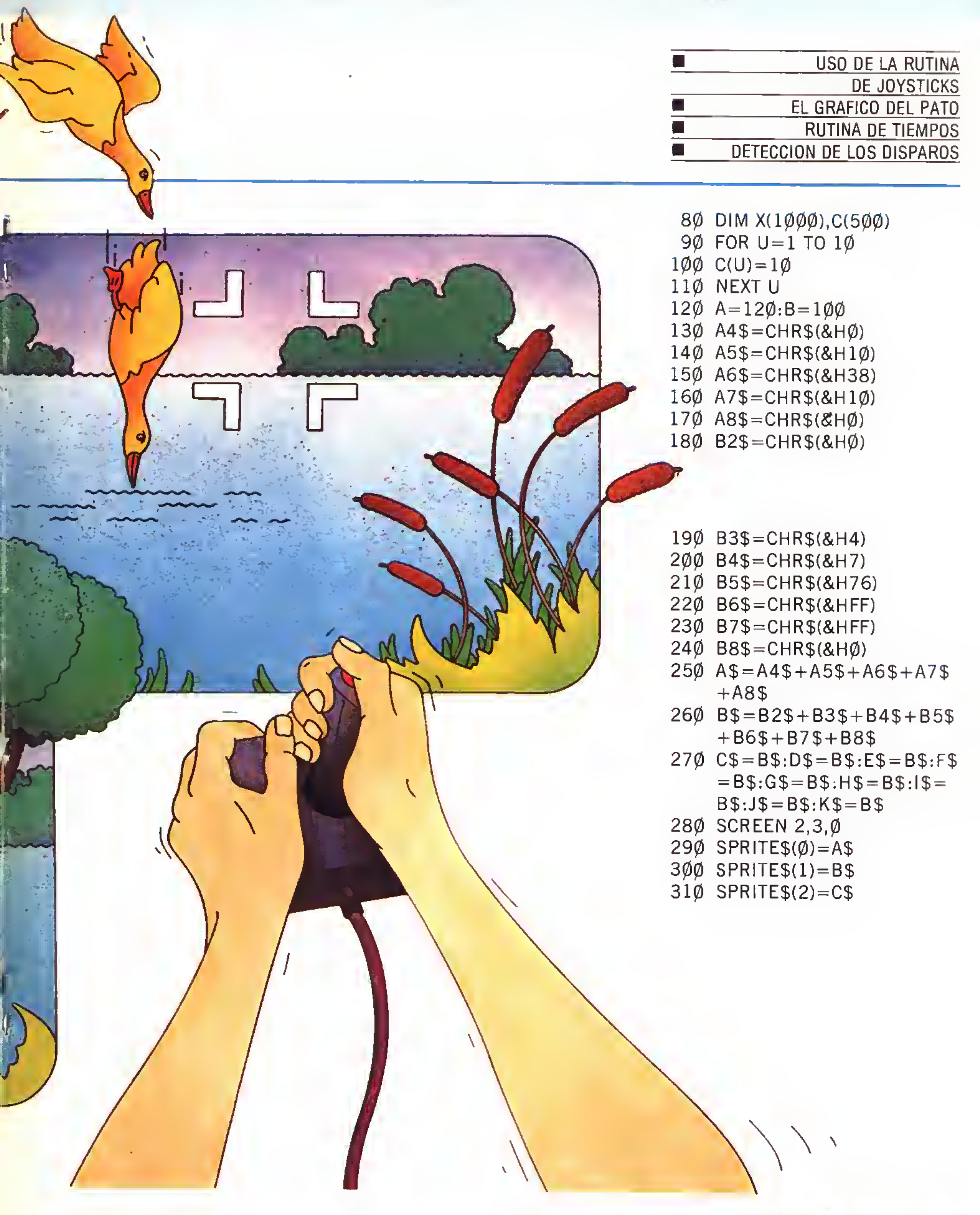
El objetivo del juego es abatir diez patos que aparecen en un corto intervalo de tiempo en posiciones aleatorias de la pantalla. El tanteo se basa en tu habilidad. Obtienes puntos por cada impacto, cuanto más rápido seas, más puntos obtendrás.

Además por cada tiro fallido se te quitarán puntos.

Carga en tu ordenador la rutina de *joystick* antes de teclear las nuevas líneas. A continuación añádele éstas y tendrás un programa de caza de patos con puntuación incorporada:

```
10 CLS:KEYOFF:COLOR5,1,
   1:SCREEN0,0,0
20 LOCATE8,8
30 PRINT "EL JUEGO DE LOS
   PATOS"
50 PRINT:PRINT:PRINT:PRINT:
   INPUT " ELIJA NIVEL DE
   JUEGO (1-8)";N
60 PLAY"V10L30DEGDEGCDF
   CDFDEGDEGCDFCDFFED"
70 CLS:LOCATE10,13:PRINT"
   ESPERE UN MOMENTO"
```

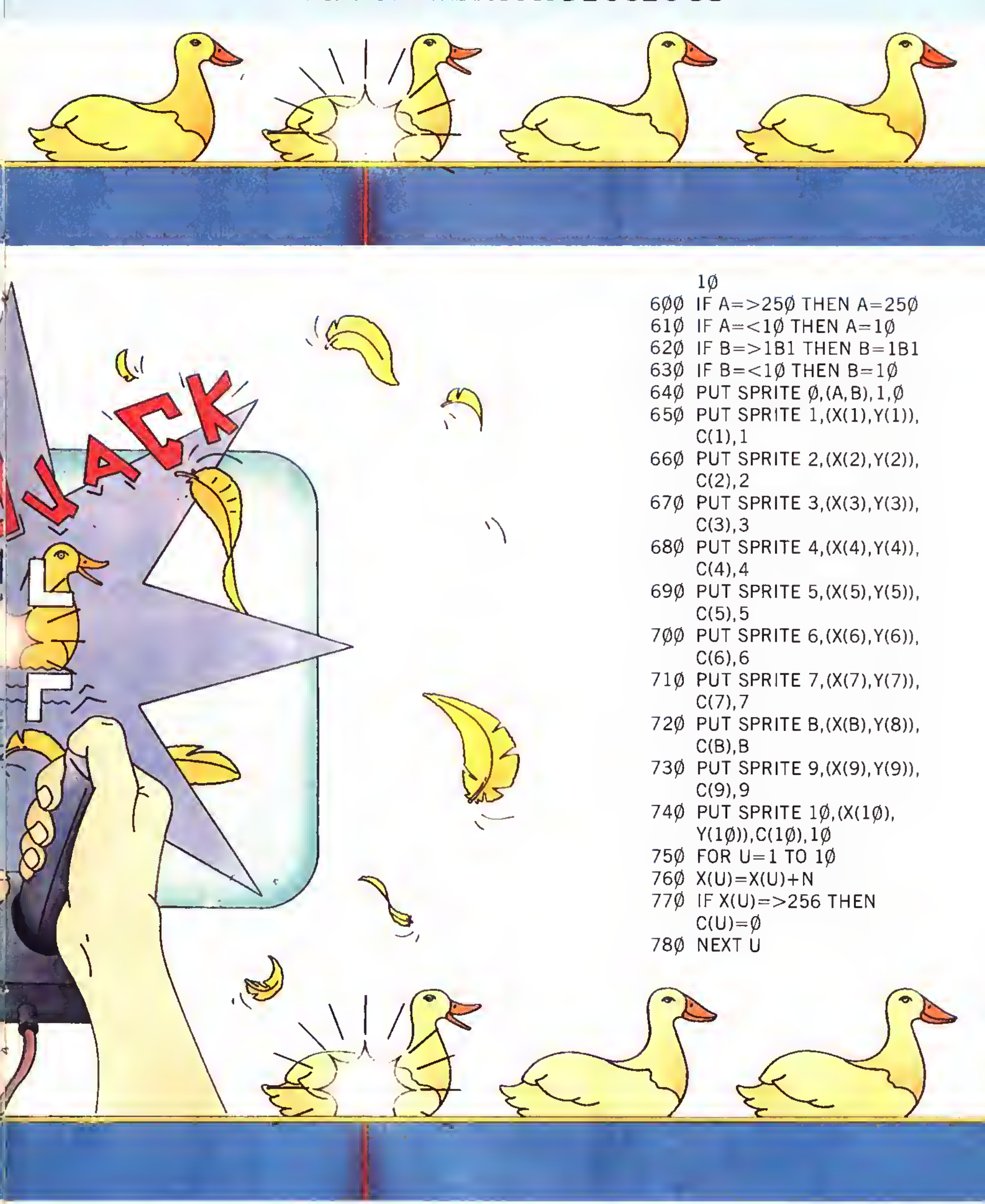
- USO DE LA RUTINA DE JOYSTICKS
- EL GRAFICO DEL PATO
- RUTINA DE TIEMPOS
- DETECCION DE LOS DISPAROS



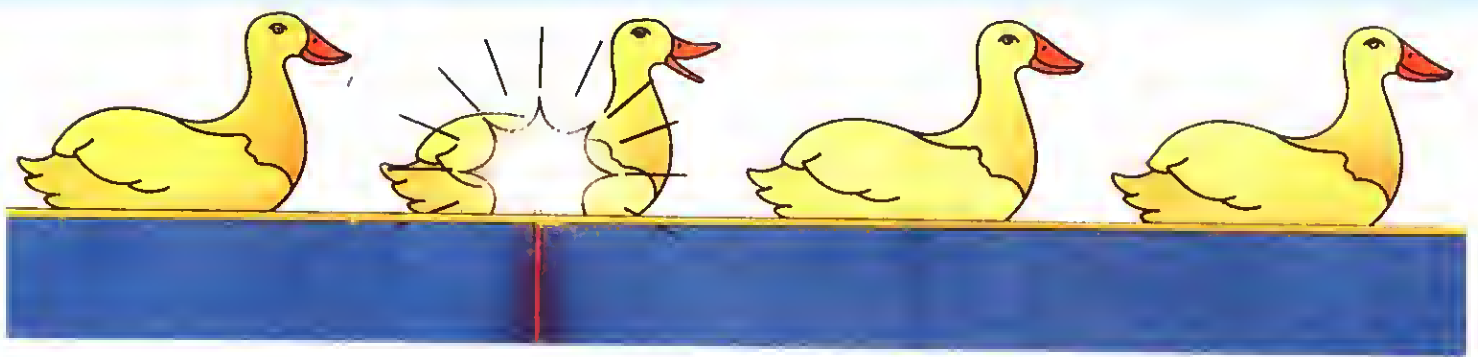
```
80 DIM X(1000),C(500)
90 FOR U=1 TO 10
100 C(U)=10
110 NEXT U
120 A=120:B=100
130 A4$=CHR$(&H0)
140 A5$=CHR$(&H10)
150 A6$=CHR$(&H38)
160 A7$=CHR$(&H10)
170 A8$=CHR$(&H0)
180 B2$=CHR$(&H0)
190 B3$=CHR$(&H4)
200 B4$=CHR$(&H7)
210 B5$=CHR$(&H76)
220 B6$=CHR$(&HFF)
230 B7$=CHR$(&HFF)
240 B8$=CHR$(&H0)
250 A$=A4$+A5$+A6$+A7$
    +A8$
260 B$=B2$+B3$+B4$+B5$
    +B6$+B7$+B8$
270 C$=B$:D$=B$:E$=B$:F$
    =B$:G$=B$:H$=B$:I$=
    B$:J$=B$:K$=B$
280 SCREEN 2,3,0
290 SPRITE$(0)=A$
300 SPRITE$(1)=B$
310 SPRITE$(2)=C$
```



```
320 SPRITE$(3)=D$
330 SPRITE$(4)=E$
340 SPRITE$(5)=F$
350 SPRITE$(6)=G$
360 SPRITE$(7)=H$
370 SPRITE$(8)=I$
380 SPRITE$(9)=J$
390 SPRITE$(10)=K$
400 VU=50
410 FORU=1TO10
420 X(U)=INT(RND(-TIME)*100)
430 Y(U)=VU:VU=VU+10
440 NEXT U
460 CLS
470 LINE (0,0)-(256,45),7,BF
480 CIRCLE (150,40),20,11,,,
    1.4
490 PAINT (150,40),11
500 LINE (0,45)-(256,191),5,BF
510 S=STICK(0)
520 IF S=1 THEN B=B+10
530 IF S=2 THEN
    A=A+15:B=B-10
540 IF S=3 THEN A=A+15
550 IF S=4 THEN
    A=A+15:B=B+10
560 IF S=5 THEN B=B+10
570 IF S=6 THEN A=A-
    15:B=B+10
580 IF S=7 THEN A=A-15
590 IF S=8 THEN A=A-15:B=B-
```




```
     1Ø
6ØØ  IF A=>25Ø THEN A=25Ø
61Ø  IF A=<1Ø THEN A=1Ø
62Ø  IF B=>1B1 THEN B=1B1
63Ø  IF B=<1Ø THEN B=1Ø
64Ø  PUT SPRITE Ø,(A,B),1,Ø
65Ø  PUT SPRITE 1,(X(1),Y(1)),
     C(1),1
66Ø  PUT SPRITE 2,(X(2),Y(2)),
     C(2),2
67Ø  PUT SPRITE 3,(X(3),Y(3)),
     C(3),3
68Ø  PUT SPRITE 4,(X(4),Y(4)),
     C(4),4
69Ø  PUT SPRITE 5,(X(5),Y(5)),
     C(5),5
7ØØ  PUT SPRITE 6,(X(6),Y(6)),
     C(6),6
71Ø  PUT SPRITE 7,(X(7),Y(7)),
     C(7),7
72Ø  PUT SPRITE B,(X(B),Y(8)),
     C(B),B
73Ø  PUT SPRITE 9,(X(9),Y(9)),
     C(9),9
74Ø  PUT SPRITE 1Ø,(X(1Ø),
     Y(1Ø)),C(1Ø),1Ø
75Ø  FOR U=1 TO 1Ø
76Ø  X(U)=X(U)+N
77Ø  IF X(U)=>256 THEN
     C(U)=Ø
78Ø  NEXT U
```



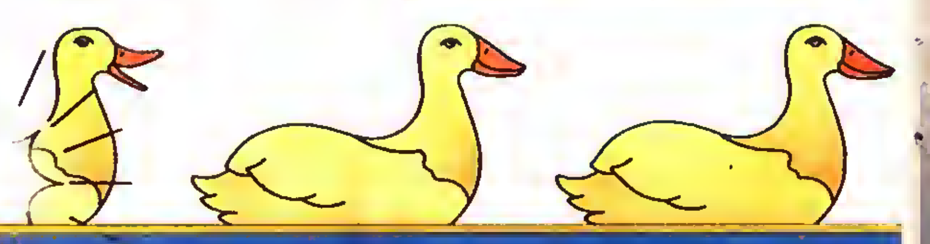
```
790 IFC(1)=ØANDC(2)=ØANDC
    (3)=ØANDC(4)=ØANDC(5)
    =ØANDC(6)=ØANDC(7)=Ø
    ANDC(8)=ØANDC(9)=ØAN
    DC(1Ø)=ØTHENGOTO99Ø
8ØØ SPRITE ON
81Ø ON SPRITE GOSUB 85Ø
82Ø STRIG(Ø)ON
83Ø ON STRIG GOSUB 1Ø8Ø
84Ø GOTO 51Ø
85Ø STRIG(Ø)ON
86Ø ON STRIG GOSUB 89Ø
87Ø GOTO 84Ø
89Ø SPRITE ON
9ØØ ON SPRITE GOSUB 92Ø
91Ø GOTO 8ØØ
92Ø UH=1:BEEP
93Ø FOR U=5Ø TO 14Ø STEP 1Ø
94Ø IF B=U THEN C(UH)=Ø
95Ø UH=UH+1
96Ø NEXT U
97Ø PA=PA+1ØØØ*N
98Ø GOTO 8ØØ
99Ø SCREEN Ø,Ø,Ø
1ØØØ COLOR 8
1Ø1Ø LOCATE6,1Ø
1Ø2Ø PRINT "PUNTUACION:";PA
1Ø4Ø LOCATE 3,15:INPUT
     "OTRA VEZ?.. S/N";PR$
1Ø5Ø IF PR$="S" OR PR$="s"
     THEN GOTO 9Ø ELSE
     1Ø6Ø
1Ø6Ø CLS:KEYOFF:LOCATE 1Ø,
     1Ø:PRINT"ADIOS
     PARDILLOS"
1Ø7Ø END
1Ø8Ø BEEP
1Ø9Ø RETURN
```

De la línea 20 a la 70 se crea una presentación en la que se nos preguntará el nivel de dificultad entre 1-8, con el que queremos jugar. La línea 80 dimensiona 2 variables que nos serán útiles más tarde. De la línea 90 a 110 se asigna el color a cada uno de los patos, mientras que desde la línea 180 a 240 se diseña un sprite con un sencillo dibujo de un pato. Desde la línea 300 a la 390 se asignan los números de sprites a cada uno de los patos. De la 400 a 450 se asignan las posiciones aleatorias de los patos en la pantalla. Las líneas comprendidas entre 650 y 740 se encargan de representar los diez patos en la pantalla. Desde la línea 750 a 790 se consigue que los patos avancen a una determinada velocidad y que vayan desapareciendo conforme llegan al final de la pantalla. Las líneas 800 y 810 son las encargadas de detectar si el sprite del punto de mira choca con el de algún pato; si esto ocurre el control pasa a la línea 850 y 860, donde se comprueba si se tiene pulsado el botón de disparo; esto se hace con el comando STRIG. El valor entre paréntesis selecciona el botón disparador; éste será la barra espaciadora cuando el valor sea Ø; el botón disparador número 1 del *joystick* cuando el valor sea de 1 o 3, y el botón disparador número 2 del *joystick* cuando los valores sean 2 o 4. Si no se tiene pulsado el botón de disparo el control retorna al programa; si, por el contrario, lo tenemos pulsado, el control pasa a las líneas 890 y 900, donde se comprueba si el sprite del punto de mira aún está encima de alguno de los sprites de los patos. Si esto no es así el control retorna al programa; caso contrario se ponen en acción las líneas comprendidas entre 920 y 980, donde se produce el sonido BEEP del disparo, se localiza el pato que ha sido alcanzado y a éste se le atribuye el color Ø para que desaparezca. Después el control vuelve al programa. Es entonces cuando empieza una subrutina de fin de programa que se extiende desde la línea 990 hasta la línea 1070. Esta subrutina es llamada desde la línea 790.

P y R

¿Qué hay que hacer para cambiar el gráfico del pato por un blanco diferente?

Simplemente hay que alterar los valores de las variables de las líneas 180 a 240, y para variar el color basta alterar la línea 1ØØ.



BARAJA Y REPARTE

- DISPOSICION DE LAS CARTAS
- DIBUJANDO LAS CARTAS
- BARAJADO
- REPARTO

Los ordenadores pueden ser muy buenos jugadores de cartas si se programan correctamente; además nunca se aburren.

Aquí tienes la forma de programar los gráficos de una baraja.

¿Te encuentras distanciado de tus amigos, parientes o colegas por haberles dejado sin un duro jugando a las cartas? ¿Eres tú el que está sin blanca por haber jugado con expertos? Sea como fuere en los capítulos siguientes de nuestro coleccionable te presentamos la solución. Programando tu ordenador para que juegue contigo a las veintiuna, tendrás una víctima propiciatoria.

En esta primera parte nos ocuparemos de la manera de generar las rutinas gráficas con las que se construye la baraja. el resto del programa que es el juego propiamente dicho se presentará en los dos capítulos siguientes.

Pero no te olvides de almacenar cada una de las secciones en cinta a medida que construyes el juego.

Si no eres un experto en el juego de las veintiuna, no te preocupes: en la última parte del programa presentaremos un conjunto completo de las reglas del juego pero antes debes ser capaz de programar un mazo de cartas.

La rutina que te permitirá barajar las cartas y preparar las rutinas gráficas sería algo así:

```
10 CLEAR 400
20 DIM C(52),P$(52),N(52),
   V$(52),W(52)
30 OPEN "grp:" AS #1
40 REM CRUPI EL CRUPIER
50 CLS:KEYOFF:COLOR 3,1,1
130 COLOR 3,1,1
150 SCREEN 2
160 LINE (10,10)-(245,150),
    10,B
170 PAINT (5,5),10
180 LINE (10,155)-(245,190),
    1,BF
190 PSET (17,160),1
200 PRINT #1,"Barajando."
210 COLOR 1
220 PSET (100,2),10
230 PRINT #1,"PUNTUACION:"
240 COLOR 3
250 D1=20:D2=15
260 RESTORE 2170
270 FOR A=1 TO 52
280 READ S
290 N(A)=S
300 READ S$
310 V$(A)=S$
320 NEXT A
330 REM BARAJANDO LA
    BARAJA
340 FOR A=1TO52
350 D=INT(RND(-TIME)*52)+1
360 FOR U=1 TO A
370 IF W(D)=53THEN GOTO
    350
380 NEXT U
390 C(A)=N(D)
400 W(D)=53
420 NEXT A
430 REM BARAJANDO PALOS
440 FOR A=1 TO 52
450 D=INT(RND(-TIME)*52)+1
460 FOR U=1 TO A
470 IF W(D)=54 THEN GOTO
    450
480 NEXT U
490 P$(A)=V$(D)
500 W(D)=54
520 NEXT A
560 LINE (10,155)-(245,190),
    1,BF
600 FOR A=1 TO 52 STEP 2
610 IF P$(A)="C" THEN GOTO
    2210
620 IF P$(A)="D" THEN GOTO
    2310
630 IF P$(A)="P" THEN GOTO
    2390
640 IF P$(A)="T" THEN GOTO
    2490
810 REM REPRESENTACION DE
    CARTA
820 COLOR 1
830 IF C(A)<=10 THEN GOTO
    900
840 IF C(A)=11 THEN VS$="J"
850 IF C(A)=12 THEN VS$="Q"
860 IF C(A)=13 THEN VS$="K"
870 IF C(A)=14 THEN VS$="A"
880 PSET (D1,D2+2),15:PRINT
    #1,VS$
890 GOTO 910
900 PSET (D1,D2+2),15:PRINT
    #1,C(A)
950 D1=D1+32:IF D1=>230
    THEN D1=10:D2=D2+55
2170 DATA 2,C,3,C,4,C,5,C,6,
     C,7,C,8,C,9,C,10,C,11,C,
     12,C,13,C,14,C
2180 DATA 2,D,3,D,4,D,5,D,6,
     D,7,D,8,D,9,D,10,D,11,D,
     12,D,13,D,14,D
2190 DATA 2,P,3,P,4,P,5,P,6,
     P,7,P,8,P,9,P,10,P,11,P,
     12,P,13,P,14,P
2200 DATA 2,T,3,T,4,T,5,T,6,
     T,7,T,8,T,9,T,10,T,11,T,
     12,T,13,T,14,T
2210 REM CARTAS
2220 REM CORAZONES
2230 LINE (D1,D2)-(D1+30,
     D2+50),15,BF
2240 CIRCLE (D1+12,D2+20),
     3,8
2250 CIRCLE (D1+17,D2+20),
     3,8
2260 PAINT (D1+12,D2+20),
     8:PAINT(D1+17,D2+20),8
```

```
2270 LINE (D1+9,
     D2+20)-(D1+15,
     D2+32),8
2280 LINE (D1+20,
     D2+20)-(D1+15,
     D2+32),8
2290 PAINT (D1+15,D2+30),8
2300 GOTO 620
2310 REM DIAMANTES
2320 LINE (D1,D2)-(D1+30,
     D2+50),15,BF
2330 LINE (D1+15,
     D2+20)-(D1+20,
     D2+25),8
2340 LINE (D1+20,
     D2+25)-(D1+15,
     D2+30),8
2350 LINE (D1+15,
     D2+30)-(D1+10,
     D2+25),8
2360 LINE (D1+10,
     D2+25)-(D1+15,
     D2+20),8
2370 PAINT (D1+15,D2+25),8
2380 GOTO 630
2390 REM PIGS
2400 LINE (D1,D2)-(D1+30,
     D2+50),15,BF
2410 CIRCLE (D1+12,D2+30),
     3,1
2420 CIRCLE (D1+18,D2+30),
     3,1
2430 PAINT (D1+12,D2+30),
     1:PAINT(D1+18,D2+30),1
2440 LINE (D1+9,
     D2+30)-(D1+15,
     D2+20),1
2450 LINE (D1+21,
     D2+30)-(D1+15,
     D2+20),1
2460 PAINT (D1+15,D2+25),1
2470 LINE (D1+15,
     D2+30)-(D1+15,
     D2+36),1
2480 GOTO 640
2490 REM TREBOLES
2500 LINE (D1,D2)-(D1+30,
     D2+50),15,BF
2510 CIRCLE (D1+15,D2+20),
     3,1
2520 CIRCLE (D1+12,D2+26),
     3,1
2530 CIRCLE (D1+18,D2+26),
     3,1
2540 PAINT  (D1+15,D2+20),1:
     PAINT(D1+12,D2+26),1:
     PAINT(D1+18,D2+26),1
2550 LINE (D1+15,D2+26)-
     (D1+15,D2+34),1
2560 GOTO 650
```



La forma de trabajo del programa es la siguiente:

La línea 20 dimensiona cinco variables: C encargada de almacenar las cartas barajadas, p$ encargada de almacenar los palos de las cartas, la N y V$ realizan la misma función que C y P$ pero sólo cuando la baraja esté ya ordenada y W es un simple contador encargado de que no existan dos cartas iguales. La línea 30 nos abre un fichero con el que podemos escribir caracteres en una pantalla en modo de gráficos. La línea 50 elimina las teclas de función y borra la pantalla. La 130 prepara el color, mientras la 150 prepara la pantalla en modo de gráficos de alta resolución. Hasta 240 se diseña el aspecto que tendrá la pantalla durante el juego donde aparecerá el mensaje BARAJANDO. La línea 250 define dos variables D1 y D2 encargadas de seleccionar el lugar de la pantalla donde aparecerá cada carta. En la 260 se indica a qué línea deben referirse las sentencias READ para leer los DATAS. Desde la 270 hasta la 320 se ordena una baraja completa de 52 cartas, de arriba abajo por corazones, diamantes, picas y tréboles. Desde la 330 hasta la 420 se baraja el mazo sin considerar los palos, es decir, en una matriz de 52 se reparten cuatro ases, cuatro dieces y así sucesivamente hasta completar la baraja. La variable W se encarga de que no coincidan dos cartas en la misma posición de la baraja y de que no haya dos iguales. En la línea 390 la carta barajada se archiva en la matriz C. Desde 430 a 520 se barajan sólo los palos, el funcionamiento de esta rutina es igual a la anterior, el palo barajado se archiva en la matriz p$. En 560 se traza un rectángulo de color negro que borra la palabra BARAJANDO que se encuentra en pantalla. La 600 crea un bucle con cabida para 26 peticiones de carta. Desde 610 a 640 se comprueba qué palo es la última carta pedida y se manda el control a la rutina que dibuja la carta correspondiente. Desde la línea 810 hasta la 910 se programa la representación del símbolo de la carta encima de su dibujo, si el símbolo de la carta está entre dos y diez es representado por la línea 900, si se encuentra entre once y catorce será sustituido por los símbolos J, Q, K o A y representado en la línea 880. La línea 950 se encarga de desplazar la posición de la pantalla donde se representará la siguiente carta. Entre 2170 y 2200 se define una baraja, desde 2210 hasta 2560 se programan cuatro rutinas que dibujan los distintos palos.



EMPIEZA EL JUEGO

- REPARTIENOO LAS CARTAS
- LAS APUESTAS
- DOBLAR, CAMBIAR Y PLANTARSE
- HACER SALTAR LA BANCA
- EJECUCION OE LOS TOTALES

El banquero fija en ti su mirada de hielo. Haces tu apuesta y recibes otra carta, pero ¿pides carta o te plantas? En esta ocasión nos ocuparemos de las líneas de programa correspondientes a la parte del jugador.

Continuando a partir de la rutina de gráficos que tecleaste en el capítulo anterior, ahora te hacen falta dos secciones más de programa, una para manejar las respuestas del jugador y otra para hacer posible que juegue el ordenador. En este capítulo veremos las lineas que hacen falta para el jugador y para las fichas, pero no llegaremos muy lejos jugando ahora, ya que todavía no has enseñado a jugar a tu ordenador.

Esta sección del programa está relacionada con unas cuantas tareas diferentes. No te preocupes si no estás muy seguro de las reglas exactas del juego de las veintiuna, ya nos ocuparemos de ellas junto con la última parte del programa. Básicamente el programa tiene que hacer tres cosas, ocuparse del reparto de las cartas, permitir al jugador hacer apuestas, y pedir cartas adicionales y finalmente tiene que calcular la puntuación del jugador.

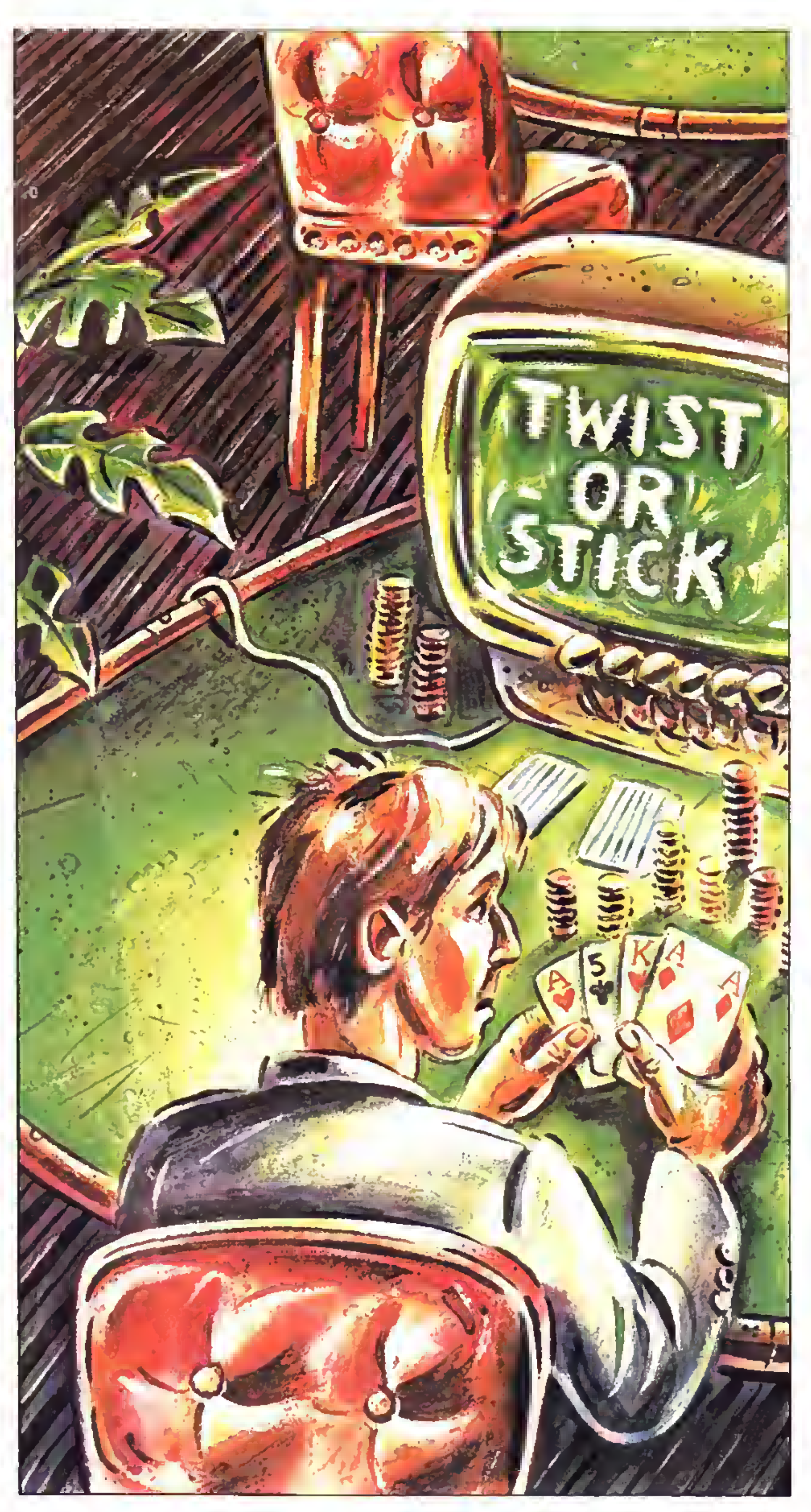
La siguiente sección del programa contabiliza los puntos del jugador, comprueba que no se haya pasado de veintiuno, o que sea igual a veintiuno o que esté entre veintiuno y dieciséis para ofrecer la posibilidad de plantarse. Después el ordenador pide al jugador que haga su apuesta por esta carta; tras hacerlo el ordenador muestra la cantidad de dinero apostado sobre la mesa, la cantidad que le queda al jugador y la que le queda a la banca. El programa además presenta mensajes al jugador dependiendo de la jugada, como: has hecho 21, te ha salido un as, etc...



Agrega ya las siguientes líneas al programa del capítulo anterior.

```
650 FOR B=2 TO 10
660 IF C(A)=B THEN PA=
    PA+B
670 NEXT B
710 IF C(A)=11 THEN
    PA=PA+10
720 IF C(A)=12 THEN
    PA=PA+10
730 IF C(A)=13 THEN
    PA=PA+10
780 IF C(A)=14 THEN GOTO
    2080
910 LINE (200,0)-(255,10),10,
    BF
920 PSET (200,2),10
930 PRINT #1,PA
940 COLOR 3
960 IF PA>21 THEN GOTO
    1240
980 IF PA=21 THEN GOTO
    1440
1000 IF PA=>16 AND PA<21
     THEN 1640
1020 LINE (10,155)-(245,
     190),1,BF
1030 PSET (17,160),1
1040 PRINT #1,
     "APUESTA."
1050 PSET (17,170),1
1060 PRINT #1,"1-100$,
     3-1000$"
1070 PSET (17,180),1
1080 PRINT #1,"2-500$,
     4-10000$"
1090 J$=INKEY$
1100 IF J$="1" THEN
     JA=JA+100:
     GOTO1150
1110 IF J$="2" THEN
     JA=JA+500:
     GOTO1150
1120 IF J$="3" THEN
     JA=JA+1000:
     GOTO1150
1130 IF J$="4" THEN
     JA=JA+10000:
     GOTO1150
1140 GOTO 1090
1150 LINE (15,130)-(230,
     148),1,BF
1160 PSET (15,140),12
1170 PRINT #1,JA*2;"$H";
     AP-JA;"$M-";
     PP-JA
1180 LINE (10,155)-(245,
     190),1,BF
1190 PSET (17,160),1
1200 PRINT #1,"CUBRO LA
     APUESTA."
1210 FOR WE=1 TO 500:NEXT
     WE
1220 NEXT A
1240 LINE (10,155)-(245,
     190),1,BF
1250 PSET (17,160),1
1260 PRINT #1,"TE HAS
     PASADO,HAS HECHO:";
      PA
1270 PSET (17,170),1
1280 PRINT #1,"YO GANO."
1290 FOR WE=1 TO 500:NEXT
     WE
1300 PP=PP+JA:AP=
     AP-JA
1310 IF PP=<0 THEN GOTO
     2570
1320 GOTO 1780
1440 LINE (10,155)-(245,
     190),1,BF
1450 PSET (17,160),1
1460 PRINT #1,"HAS HECHO
     21."
1470 PSET (17,170),1
1480 PRINT #1,"TU GANAS."
1490 FOR WE=1 TO 500:NEXT
     WE
1500 AP=AP+JA:PP=PP-JA
1510 IF PP=<0 THEN GOTO
     2570
1520 GOTO 1780
1640 LINE (10,155)-(245,
     190),1,BF
1650 PSET (17,160),1
1660 PRINT #1,
```






```
      "(1)PLANTARSE."
1670 PSET (17,170),1
1680 PRINT #1,"(2)PEDIR
     CARTA."
1690 K$=INKEY$
1700 IF K$="1" THEN GOTO
     1880
1710 IF K$="2" THEN GOTO
     1010
1720 GOTO 1690
1780 LINE (10,155)-(245,
     190),1,BF
1790 PSET (17,160),1
1800 PRINT #1,"(1)VOLVER A
     JUGAR."
1810 PSET (17,170),1
1820 PRINT #1,
     "(2)RETIRARTE."
1830 K$=INKEY$
1840 IF K$="1" THEN GOTO
     130
1850 IF K$="2" THEN END
1860 GOTO 1830
1880 LINE (10,155)-(245,
     190),1,8F
1890 PSET (17,160),1
1900 PRINT #1,"TE PLANTAS
     Y"
1910 IF PA>PB THEN PSET (17,
     170),1:PRINT #1,
     "GANAS."; PA;"a";
      PB:FOR WE=1 TO
     500:NEXT WE:AP=AP+
     JA:PP=PP-JA:GOTO 1950
1920 IF PB>PA THEN PSET (17,
     170),1:PRINT #1,
     "PIERDES."; PB; "a";
      PA:FOR WE=1 TO
     500:NEXT WE:AP=AP-
     JA:PP=PP+JA:GOTO 1950
1930 FOR WE=1 TO 500:NEXT
     WE
1940 PSET (17,170),1:PRINT
     #1,"HEMOS
     EMPATADO.":FOR WE=1
     TO 500:NEXT WE
1950 IF PP=<0 THEN GOTO
     2570
1960 GOTO 1780
2080 LINE (10,155)-(245,
     190),1,BF
2090 PSET (17,160),1
```

```
2100 PRINT #1,"TE HA SALIDO
     UN AS"
2110 PSET (17,170),1
2120 PRINT #1,"(1)SUMAR
     1-(2)SUMAR 11"
2130 K$=INKEY$
2140 IF K$="1" OR K$="2"
     THEN GOTO 2150 ELSE
     2130
2150 IF K$="1" THEN
     PA=PA+1 ELSE
     PA=PA+11
2160 GOTO 790
```

MÁS CARTAS

Las lineas 650 a 670 comprueban si la carta del jugador no es una figura, en cuyo caso la variable PA es sumada al valor original de la carta. Desde 710 a 730 se comprueba si la carta es una figura, sumando entonces 10 a PA. La linea 780 comprueba si es un as, en cuyo caso manda el control a una rutina que nos pregunta si queremos sumar 1 u 11. Desde la linea 910 a 940 se imprime la puntuación total en la pantalla y la 960 se cerciora de que no nos hayamos pasado de 21. De haberlo hecho el ordenador contestaría: TE HAS PASADO, YO GANO. La línea 980 comprueba si hemos hecho 21, en cuyo caso nos diria HAS HECHO 21, TU GANAS; en caso de que nuestros puntos superen 16 la línea 1000 pasa el control a una rutina que nos pregunta si queremos plantarnos o pedir carta.

APUESTAS

Entre 1020 y 1140 se extiende una rutina que nos pregunta la cantidad que queremos apostar, esta cantidad es guardada carta tras carta por la variable JA, en 1170 se nos muestra en pantalla la cantidad de dinero apostado sobre la mesa, la cantidad en disposición del jugador y la cantidad en disposición de la máquina, el resto de esta parte del programa está formado por subrutinas de mensajes como, por ejemplo, volver a jugar o retirarte, te plantas y ganas, te plantas y pierdes.



HACER SALTAR LA BANCA (I)

En esta oportunidad trataremos de convertír en realidad el sueño que calladamente acaricia todo jugador: hacer saltar la banca.

La cantidad de dinero que le resta a la máquina es igual a la variable PP, que almacena todo su dinero, menos JA que es igual a la cantidad apostada. Si la cantidad de dinero que le resta a la banca es igual a cero, será detectado en las rutinas de mensajes y se enviará el control a la línea 2570 donde aparecerá un mensaje de felicitación y habremos ganado; el control retorna entonces al principio, preguntándonos de cuánto dinero disponemos.

Para completar el juego de las veintiuna tienes que programar tu ordenador para que juegue cuando le toque el turno. Aquí tienes, además, una descripción de las reglas del juego para el caso de que no estés familiarizado con el mismo.

LAS REGLAS DEL JUEGO

Antes de pasar a examinar el resto de la programación, vale la pena recapitular las reglas del juego.

El juego de las veintiuna se juega con un mazo ordinario de 52 cartas. Las cartas del 2 al 10 valen lo que indica su número; las figuras valen 10, y el as vale 1 u 11 según las necesidades del jugador. En este juego el ordenador se ocupa de ir llevando la cuenta de los totales, por lo que no tienes que hacerlo tú.

Normalmente se juega con dinero, o con «piedras», pero en esta versión para ordenador hay que programarle para que cuente la puntuación en fichas imaginarias; con las otras fichas un programador poco escrupuloso podría hacer que la máquina dejara de ser honrada.

El ordenador se programa para actuar como banquero en todo momento y será siempre el encargado de repartir las cartas.

Al principio del juego se barajan las cartas y se sacan dos de ellas poniéndolas boca abajo. En la pantalla aparecerá boca arriba la carta del jugador, aunque el programa ha sido diseñado para que la máquina no sepa qué carta es la que tiene el jugador.

El jugador debe apostar ahora sobre esta primera carta, antes de que se saque una nueva para cada uno.

El objeto del juego es terminar con mejor puntuación que la banca, es decir con un valor total más elevado. Una mano que sume en total más de 21 se pasa y pierde. Una mano con puntuación entre 16 y 21 solamente vence a la banca si la máquina tiene una mano más baja o se ha pasado de 21.

Las veintiuna del jugador vencen a todo lo que tenga la máquina excepto a otras veintiuna.

EL PROGRAMA

En primer lugar el ordenador se presenta y nos pregunta de qué cantidad de dinero disponemos. Tras esto baraja las cartas y reparte una al jugador y otra para él mismo, nos indica la puntuación total hasta el momento y nos pide que hagamos una apuesta por esa carta. En la parte inferior de la pantalla, aparece un menú con opciones para apostar entre 100 y 10.000 dólares. Presionando el número seguido de un guión que aparece al lado de cada cifra elegiremos la apuesta. Una vez hecho esto, el programa nos mostrará la cantidad de dinero apostado que se encuentra sobre la mesa, el dinero que nos queda y la cantidad que le resta a la máquina.



Tras el mensaje de «CUBRO LA APUESTA» aparece una indicación; dependiendo de la jugada, los mensajes posibles de la máquina pueden ser:

—«TE HAS PASADO»; «YO GANO», en caso de que nos pasemos de 21.

—«ME HE PASADO, TU GANAS», en caso de que la máquina se pase de 21.

—«HAS HECHO 21, TU GANAS».

—«HE HECHO 21, YO GANO».

—Un menú con las opciones de plantarse o pedir carta en caso de que nuestra puntuación sea mayor de 16.

—Un menú con las opciones de volver a jugar o retirarse.

—Un mensaje de «TE PLANTAS Y GANAS» o «TE PLANTAS Y GANO».

—Un menú con la indicación «TE HA SALIDO UN AS. ¿QUIERES SUMAR UNO U ONCE?».

Una diferencia con respecto a los juegos ordinarios de cartas es que el reparto no puede pasar al jugador, debiendo ser siempre el ordenador el que «da» las cartas. En el juego ordinario la banca cambia cuando se logran las 21, en este juego la banca es siempre el ordenador, por lo que siempre tiene ventaja sobre el jugador; en las 21 ordinarias, la banca tiene también ventaja sobre el jugador. Por ello el jugador tiene ante sí la difícil tarea de hacer saltar a la banca para ganar. Para contrarrestar las ventajas de la máquina el ordenador siempre contabilizará 11 cuando tenga un «AS».



```
60 PRINT "HOLA SOY CRUPI EL
   CRUPIER DE CUANTOS
   DOLARES DISPONES.
   (CIFRAS REDONDAS)
   APUESTA MINIMA 100"
70 INPUT AP
80 IF AP>5000000# THEN
   PRINT "LA BANCA NO
   DISPONE DE TANTO
   DINERO.":GOTO 60
90 IF AP<100 THEN GOTO 60
100 PRINT "BIEN,PUEDO
    CUBRIR ESA CANTIDAD."
110 PP=AP
120 FOR WE=1 TO 1000:NEXT
    WE
680 FOR B=2 TO 10
690 IF C(A+1)=B THEN
    PB=PB+B
700 NEXT B
740 IF C(A+1)=11 THEN
    PB=PB+10
750 IF C(A+1)=12 THEN
    PB=PB+10
760 IF C(A+1)=13 THEN
    PB=PB+10
770 IF C(A+1)=14 THEN
    PB=PB+11
790 LINE (20,100)-(150,120),
    1,BF
970 IF PB>21 THEN GOTO
    1340
990 IF PB=21 THEN GOTO
1010 IF PB=>16 AND PB<21
     THEN 1740
1340 LINE (10,155)-(245,
     190),1,BF
1350 PSET (17,160),1
1360 PRINT #1,"ME HE
     PASADO,HE HECHO:"; PB
1370 PSET (17,170),1
1380 PRINT #1,"TU GANAS."
1390 FOR WE=1 TO 500:NEXT
     WE
1400 AP=AP+JA:PP=PP-JA
1410 IF PP=<0 THEN GOTO
     2570
1420 GOTO 1780
1540 LINE (10,155)-(245,
     190),1,BF
1550 PSET (17,160),1
1560 PRINT #1,"HE HECHO
     21."
1570 PSET (17,170),1
1580 PRINT #1,"YO GANO."
1590 FOR WE=1 TO 500:NEXT
     WE
1600 PP=PP+JA:AP=AP-JA
1610 IF PP=<0 THEN GOTO
     2570
1620 GOTO 1780
1740 LINE (10,155)-(245,
     190),1,BF
1750 IF PB>17 THEN GOTO
     1980
1760 GOTO 1020
1980 LINE (10,155)-(245,
     190),1,BF
1990 PSET (17,160),1
2000 PRINT #1,"ME PLANTO Y'
```

```
2010 PSET (17,170),1
2020 IF PA>PB THEN PSET (17,
     170),1:PRINT #1,
     "PIERDO.";PA;"a";
     PB:FORWE=1TO500:
     NEXTWE:AP=AP+JA:
     PP=PP-JA:GOTO2050
2030 IF PB>PA THEN PSET (17,
     170),1:PRINT #1,
     "GANO.";PB;"a";
     PA:FORWE=1TO500:
     NEXTWE:PP=PP+JA:
     AP=AP-JA:GOTO2050
2040 PSET (1/,160),1:PRINT
     #1,"HEMOS
     EMPATADO.":FORWE=1
     TO 500:NEXT WE
2050 IF PP=<0 THEN GOTO
     2570
2060 GOTO 1780
2570 REM DESBANCAR A LA
     BANCA
2580 SCREEN 0,0,0
2590 PRINT "HAS DESBANCADO
     A LA BANCA."
2600 PRINT "FELICIDADES."
2610 FOR WE=1 TO 1000:NEXT
     WE
2620 GOTO 10
```

Desde las líneas 60 hasta la 120 se crea una presentación donde se nos pregunta de qué cantidad de dinero disponemos, almacenando esa cantidad en las variables PP, para el ordenador y AP, para el jugador. Desde 680 a 700 se comprueba la puntuación de la máquina archivándola en la variable PB. De las líneas 740 a 770 se comprueba la puntuación, en el caso de que la carta sea una figura. La línea 770 es la encargada de sumar 11 cuando la carta de la máquina sea un as. La 970 se asegura de que el ordenador no se haya pasado de 21. La 990 comprueba que el ordenador no haya hecho 21 y la línea 1010 lleva el control del ordenador a 1740 en caso de que su puntuación sea mayor de 16. Entre 1340 y 1420 se extiende la rutina utilizada cuando la máquina se pasa. Entre 1540 y 1620 se extiende la utilizada cuando el ordenador hace 21. De la línea 1740 a la 1760 el ordenador decide si debe plantarse o pedir carta. Las líneas 1980 y 2060 albergan la rutina utilizada cuando el ordenador se planta y, por último, entre 2570 y 2620 se encuentra el mensaje anunciando que ha saltado la banca.



SIMULADOR DE VUELO

- LA SIMULACION DE VUELO
- PROGRAMAS DE ENTRENAMIENTO
- VUELO MEDIANTE INSTRUMENTOS
- EL MOVIMIENTO DEL AVION
- PERDIDA DE VELOCIDAD

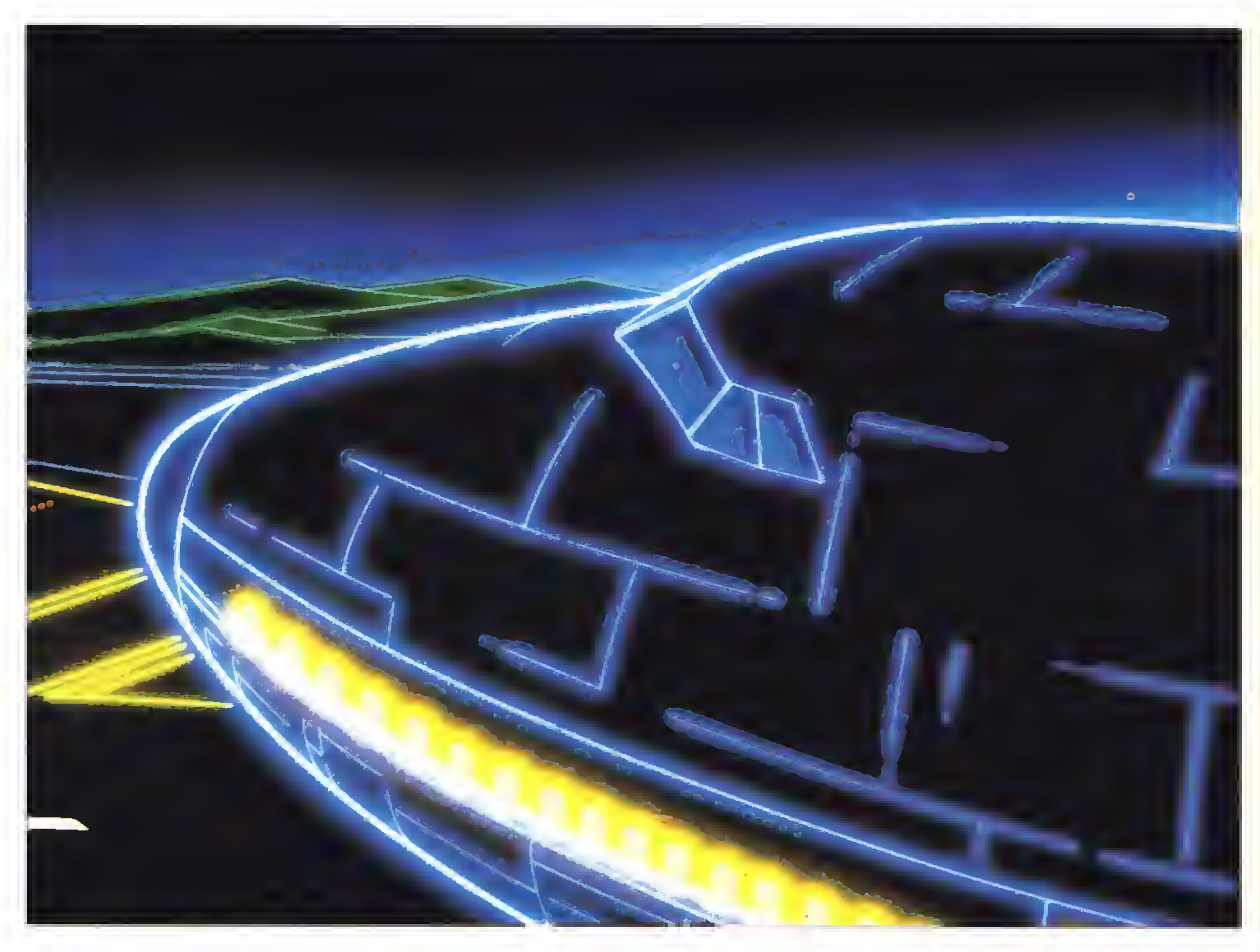
Este programa de simulación de vuelo es similar a los que se emplean en las escuelas de vuelo para enseñar a los pilotos cómo tienen que volar utilizando únicamente sus instrumentos; en la primera parte se reproduce la cabina

Los programas de juegos varían desde una fantasía desbordante que supone la entrada a mundos imaginarios y la participación en aventuras, hasta la simulación de situaciones de la vida real. Esto te permite poner a prueba tu capacidad en situaciones potencialmente peligrosas, sin tener que hacerte daño o perder totalmente millones de pesetas en costosos equipos.

Los programas de simulación de vuelo contienen un elemento de fantasía: tú solo en la cabina, con toda la tripulación aquejada de una misteriosa enfermedad, con una sola mano consigues hacer que el avión tome tierra felizmente. Pero los sofisticados programas de este tipo tienen un uso práctico real, hasta el punto de que casi todas las principales compañías de líneas aéreas y escuelas de vuelo los utilizan con regularidad.

SIMULADORES DE ENTRENAMIENTO

En el extremo superior de la escala está la simulación total, la llamada «Fase 3» en la terminología de las administraciones de aviación civil, que te permite experimentar las mismas sensaciones que un piloto en un avión de verdad. Tú ves lo mismo que él ve a



través de la ventanilla de la carlinga (incluyendo una pequeña diferencia de ángulo en el punto de vista para la posición del copiloto); sentirás lo mismo que él siente en los despegues y en los aterrizajes, así como las turbulencias; oirás lo mismo que él oye, incluyendo las indicaciones del control de tráfico aéreo. En teoría un piloto puede completar todo su entrenamiento en uno de estos simuladores y obtener su liceneia sin tener que abandonar el suelo para nada.

SIMULADORES DE SOBREMESA

En el extremo opuesto de la escala están los programas de simulación de vuelo muy parecidos al que veremos a continuación.

Las unidades de sobremesa se pueden «volar» en el interior de un aula, y resultan útiles para la enseñanza de los procedimientos de cabina y para desarrollar la rapidez de reflejos de los pilotos.

Resultan además esenciales para la enseñanza del vuelo por instrumentos, una técnica que permite al piloto navegar apoyándose únicamente en el panel de instrumentos, algo que todo piloto ha de hacer cuando las condiciones meteorológicas son malas.

LO QUE HACE EL PROGRAMA

Este artículo consta de tres partes y en él se presenta un programa de simulación de vuelo en el que se supone que te has hecho cargo del control del avión.

Tú mismo escogerás la altura y la distancia respecto del aeropuerto con la que quieres comenzar el juego. Por la ventanilla de la cabina es muy poco lo que puedes ver, solamente el horizonte, cuando hay visibilidad, y un punto distante que es la pista de aterrizaje, por lo cual, como piloto sensato que eres, tendrás que confiar en tu experiencia y atendiendo a lo que te indique el panel de instrumentos ponerte a salvo en tierra a ti y a tus pasajeros, cuya seguridad depende absolutamente de tu pericia.

LOS INSTRUMENTOS

En tu panel de instrumentos hay cinco diales. El primero te informa de la velocidad del avión. Este valor se incrementará automáticamente con sólo pulsar una vez la tecla correspondiente. Un contador, situado debajo del dial de la velocidad, te indica la orientación de tu vuelo, haciendo las veces de brújula.

Un tercer dial te muestra la altura en que te hallas en cada momento.

El cuarto dial señala la distancia a la que te encuentras del aeropuerto más próximo; ésta disminuirá más o menos rápidamente en función de la velocidad a la que estés avanzando.

El último dial es el del combustible. La cantidad que llevas en tu depósito la elegirás tú antes de comenzar el juego. Has de tener cuidado de no poner poca cantidad para evitar quedarte sin combustible durante el vuelo. Del mismo modo, llenar mucho el depósito, restará velocidad y altura a tu avión.

Si rebasas los 15.000 metros de altura, tu fuselaje no lo podrá resistir y caerás en picado.

ATERRIZAJE

Aunque hay programas de simuladores de vuelo en los que la imagen que se ve a través de la ventanilla de la cabina se va haciendo más nítida a medida que progresa la aproximación, no es éste nuestro caso. Por eso debes centrar la imagen en relación al mensaje de la torre de control que aparecerá en uno de los diales de la pantalla, cuando falten mil metros para el aterrizaje.

Para que el aterrizaje tenga un final feliz, debes hacer coincidir el valor del mensaje de la torre de control con el marcador de derivación. Cuando el indicador de distancia señale cero, estarás justo encima de la pista. Es entonces cuando debes decrementar la velocidad y la altura hasta que todas las magnitudes sean cero. Si no lo has previsto a tiempo, lo más probable es que la pista se te quede corta y te salgas de ella provocando un grave siniestro.

MOVIENDO EL AVION

El margen de los controles de que dispones se aproxima bastante a los controles de un avión de verdad, aunque estés pulsando teclas en vez de utilizar un *joystick*. En una aeronave real, el control de la elevación —los

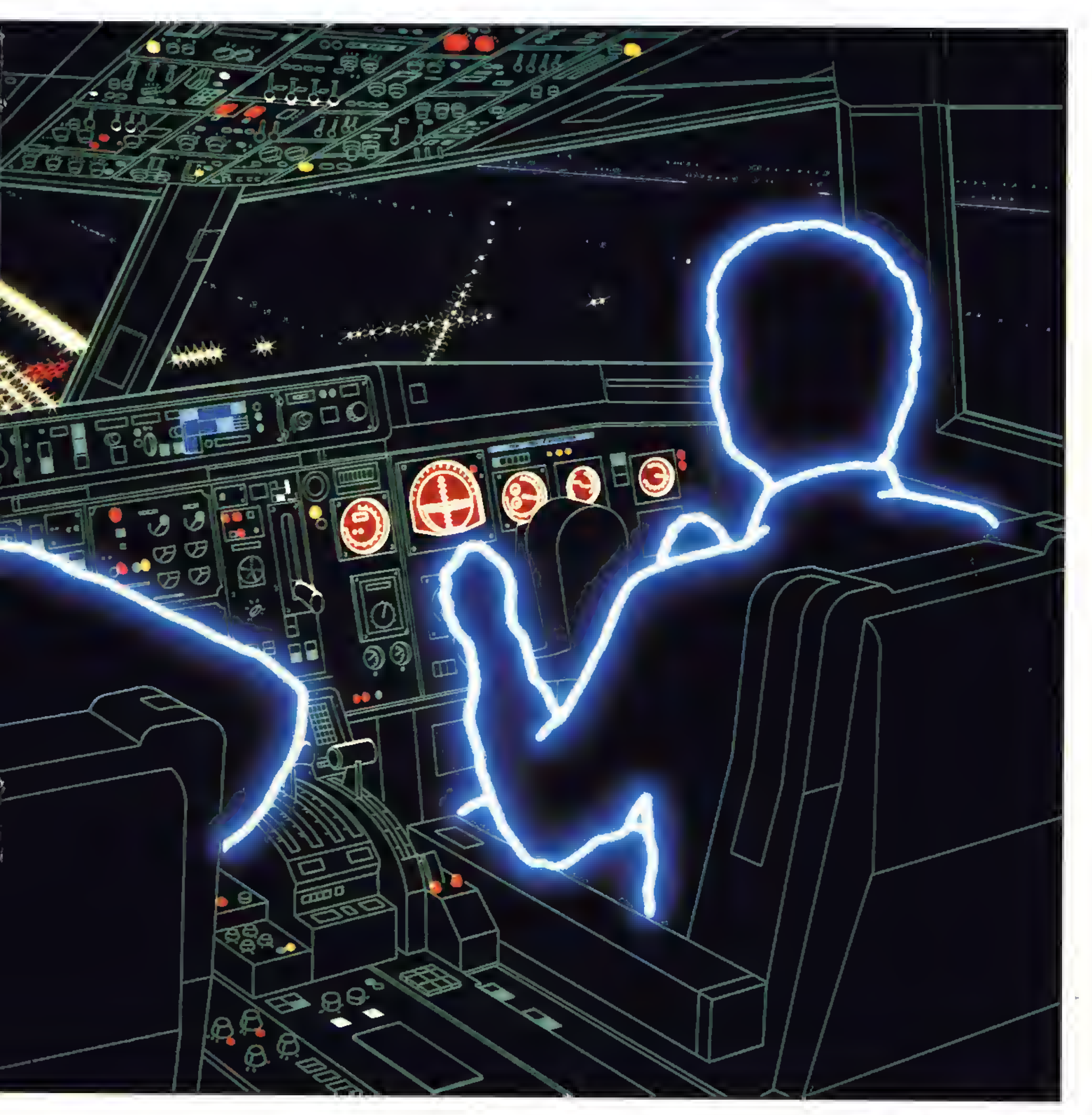


movimientos hacia arriba y hacia abajo— se hace moviendo el *joystick* hacia atrás o adelante.

Los mandos que te permitirán controlar correctamente el avión son los siguientes: la tecla «q» bajará el avión en intervalos de diez en diez metros; la tecla «a» efectuará similar movimiento pero en sentido inverso, esto es, ascendiendo; la tecla «w» efectúa la misma operación que «a» pero a intervalos de cien en cien; la tecla «s» eleva la altura del avión de cien en cien.

Para la derivación hacia la derecha debes pulsar la tecla «p», mientras que para hacer lo mismo hacia la izquierda debes pulsar la tecla «o».

La tecla «I» te permitirá elevar la velocidad de tu avión a intervalos de diez en diez, mientras que «u» lo hará de cien en cien, «L» decrementará la velocidad de diez en diez y «k» de cien en cien.



PERDIDA DE VELOCIDAD

Cuando la velocidad de un avión cae por debajo de un determinado valor se dice que entra en pérdida de velocidad y en ese momento el avión empieza a caer como si fuera una piedra. En este programa, cuando el valor de tu dial de velocidad decaiga por debajo de 100 caerás en picado contra el suelo. Una entrada en pérdida es algo que aterroriza a cualquier piloto.

DIVISION DEL PROGRAMA

El programa es demasiado largo y complejo como para darlo todo de una sola vez, por lo que lo hemos dividido en tres partes.

Lo que se hace en esta primera parte es configurar la pantalla para mostrar el interior de la cabina, con su ventanilla, los cuatro *diales* debidamente etiquetados y los letreros de los contadores.

Los comandos que intervienen serán familiares para la mayoría de vosotros por haberlos visto ya en otros programas.

La parte de programa introducida en la parte dos, hace posible que los *diales* y contadores sean sensibles al movimiento del avión y hay un comando temporal que hace que éste vuele aleatoriamente sin que haya un piloto que actúe sobre los controles, para que puedas ver funcionando el panel de instrumentos. La sección final te permite tomar el control del avión y realizar una estimación de tu técnica de aterrizaje para que puedas juzgar tus progresos.

DIBUJANDO LA CABINA

Para dibujar la cabina teclea en tu ordenador la siguiente parte del programa:

```
10 CLS:COLOR 3,1,1:KEY
   OFF
20 INPUT "CON QUE ALTURA
   QUIERES EMPEZAR:";AL
30 INPUT "A QUE DISTANCIA
   DEL AEROPUERTO:";
   DI
40 INPUT "CON CUANTO
   COMBUSTIBLE:";CO
50 IF CO>20000 THEN PRINT
   "ESA CANTIDAD NO CABE EN
   EL DEPOSITO.":GOTO
   40
60 INPUT "A QUE VELOCIDAD:";
   VE
70 INPUT "NIVEL DE
   DIFICULTAD(1-10)";D
   80DP=1100-(D*100)
90 COLOR 15,1,1
100 SOUND 1,8:SOUND 7,
    35:SOUND 11,100
110 REM FACTORES
    INICIALES
120 PP=INT(RND(-TIME)*360)
    +1
130 GR=180:CC=1
140 REM SIMULADOR DE
    VUELO
150 SCREEN 2,0,0
160 OPEN "GRP:" AS # 1
170 LINE (10,10)-(245,100),
    15,B
180 LINE (20,130)-(70,150),
    15,B
190 LINE (80,130)-(130,150),
    15,B
200 LINE (140,130)-(190,
    150),15,B
210 LINE (200,130)-(250,
    150),15,B
220 PSET (30,2),1
230 PRINT # 1,"PISTA DE:";DP;
    " METROS."
240 PSET (20,160),1:PRINT #
    1,"DERIV:"
250 PSET (20,120),1
260 PRINT # 1,"VELOC"
270 PSET (110,160),1:PRINT #
    1, "COMBUS:"
280 PSET (80,120),1
290 PRINT # 1,"ALTURA"
300 PSET (140,120),1
310 PRINT # 1,"DISTA"
320 PSET (200,120),1
330 PRINT # 1,"G.PISTA"
340 LINE (21,131)-(69,149),1,
    BF
350 PSET (21,135),1
360 PRINT # 1,VE
370 LINE (70,155)-(100,170),
    1,BF
380 PSET (70,160),1
390 PRINT # 1,GR
400 LINE (170,155)-(255,
    170),1,BF
410 PSET (170,160),1
420 PRINT # 1,CO
430 LINE (81,131)-(129,149),
    1,BF
440 PSET (81,135),1
450 PRINT # 1,AL
460 LINE (141,131)-(189,
    149),1,BF
470 PSET (141,135),1
480 PRINT # 1,DI
```

La línea 10 borra la pantalla y establece los colores del programa. Entre las líneas 20 y 70 se introducen los valores correspondientes a la altura, distancia, combustible, velocidad y el nivel de dificultad que tú elijas.

El cálculo del espacio del que dispones para aterrizar se realiza en la línea 80. Esta distancia disminuirá en función del nivel de dificultad que escojas.

El sonido que simula los reactores del avión se inicializa en la línea 100, variándolo en función de la altura y la velocidad.

En la linea 120 se elige aleatoriamente la pista en la que deberás aterrizar.

Este cálculo depende del tiempo transcurrido desde que encendiste el ordenador.

DIBUJO DEL PARABRISAS Y LOS DIALES

Los dibujos del parabrisas de la cabina y de los diales se ejecutan entre las líneas 170 a 210, mediante las instrucciones LINE.

El resto de programa se divide en dos partes. La primera, desde las líneas 230 a 330 imprime el título de los diales.

Mientras que la segunda parte, entre 340 y 480, se encarga de renovar el contenido de cada dial.

Si ejecutas ahora el programa te aparecerá en la pantalla tu cabina simulada de avión.

DESPEGA PARA TU PRIMER VUELO

- VUELO CON EL PILOTO AUTOMATICO
- APROXIMACION A LA PISTA DE ATERRIZAJE
- DIBUJANDO LA TRAYECTORIA
- EL PANEL DE INSTRUMENTOS

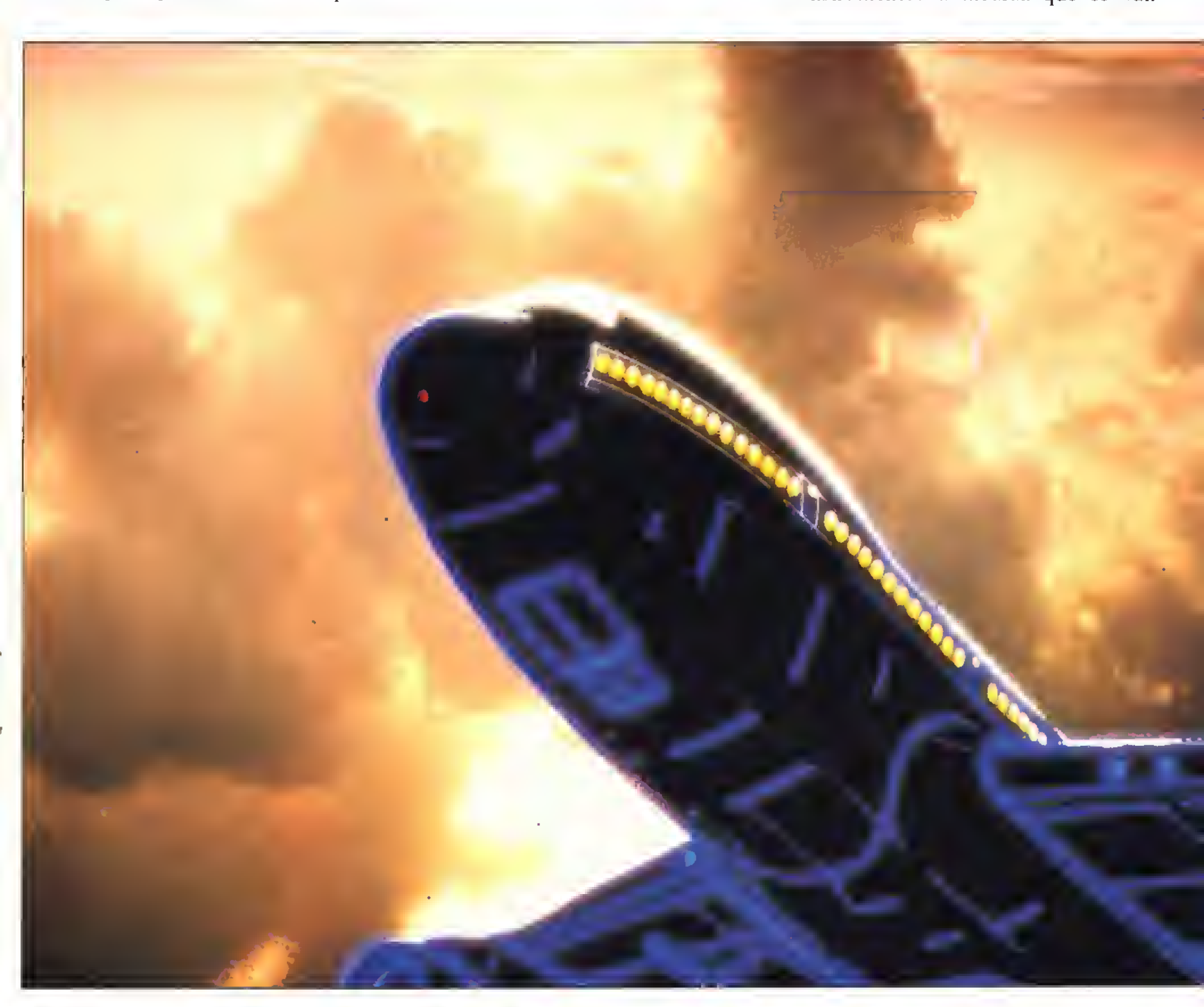
En la segunda parte del simulador de vuelo, puedes arrancar los motores y obsevar cómo adquiere vida tu panel de instrumentos. Pero ten cuidado: ¡El piloto automático se ha vuelto loco!

En la primera parte has tecleado las líneas que reproducían sobre la pantalla el interior de la cabina de vuelo.

En esta parte verás adquirir movimiento a tu avión y cómo adquiere vida el panel de instrumentos; de modo que, aunque todavía no has tomado los mandos, puedes ver cómo responde el panel de instrumentos del avión ante los movimientos del mismo.

EL AVION VUELA

En esta parte hay una larga serie de variables interdependientes que tienen que ser constantemente actualizadas para controlar el avance del avión. Además hay que volver a dibujar de modo constante el panel de instrumentos a medida que se van



modificando la posición y la altura.

La proximidad de la pista te será señalada a partir de los 1.000 metros de distancia. Uno de los diales alertará de forma intermitente sobre cuál es la pista designada desde la torre de control como lugar de aterrizaje. Debes memorizar con precisión este valor, pues a partir de los 900 metros no tendrás más información sobre él.

Durante el vuelo, no intentes virar el avión a menos de 200 metros de altura debido a que podría peligrar la vida no sólo de los viajeros, sino también de los ciudadanos que estén circulando por las calles de la ciudad o en sus rascacielos.

Para poder actualizar los diales y los contadores hay que hacer una estimación de las variables que van cambiando con arreglo a la forma en que afectan a las lecturas, después de lo cual se puede rehacer el dibujo.

Para poder dibujar con precisión la posición del avión hay que tener en cuenta muchos factores, por ejemplo, la dirección en que estás volando, la velocidad de avance, que depende en parte del peso del combustible o de la velocidad de ascenso.

```
620 AL=AL+A1
630 VE=VE+V1
640 GR=GR+G1
650 DI=DI-INT(VE/60)
660 CO=CO-INT(VE/1000)
670 REM FACTORES PESO
    COMBUSTIBLE
680 VE=VE-INT(CO/40000!)
690 AL=AL-INT(CO/40000!)
700 REM ----------
710 P1=VE/100:IF P1=>31
    THEN P1=31
720 P2=AL/200:IF P2=>15
    THEN P2=15
730 IF P1=<1 THEN P1=1
740 IF P2=<1 THEN P2=1
750 SOUND 6,P1:SOUND 8,P2
760 LINE (11,R1)-(244,R2),1
770 R1=AL*100/3000+Y1
780 R2=AL*100/3000+Y2
790 IF R1=<11 THEN R1=11
800 IF R2=<11 THEN R2=11
810 IF R1=>99 THEN R1=99
820 IF R2=>99 THEN R2=99
830 LINE (11,R1)-(244,R2),3
840 IF DI<1000 AND DI>900
    THEN GOSUB 940
850 IF DI<0 THEN GOTO 1230
860 IF AL>15000 THEN GOTO
    980
870 IF AL<0 THEN GOTO 1040
880 IF CO=<0 THEN GOTO
    1080
890 IF VE<100 THEN GOTO
    1120
900 IF VE>2500 THEN GOTO
    1150
910 IF K$="O" AND AL=<200
    THEN GOTO 1550
920 IF K$="P" AND AL=<200
    THEN GOTO 1550
930 GOTO 340
940 PSET (200,135),1
950 PRINT # 1,PP
960 LINE (201,131)-(249,
    149),1,BF
970 RETURN
```

La línea 620 realiza el descenso y ascenso automático del avión, mientras que la 630 realiza la misma función con la velocidad. La línea 640 se encarga de desplazar el avión efectuando las funciones de timón, en busca de alcanzar la orientación adecuada para poder aterrizar.

La línea 650 decrementa la distancia del aeropuerto dependiendo de la velocidad de tu aparato. La línea 660 hace disminuir el combustible de tu depósito dependiendo también de la velocidad. En la línea 680 se hace disminuir la velocidad con relación al peso del combustible. La 690 hace lo mismo que la anterior, pero variando la altura.

Las líneas 710 y 720 se encargan de que los valores de las sentencias SOUND no rebasen un valor superior a 31 y 15. Las líneas 730 y 740 realizan la misma función, pero impidiendo que su valor sea inferior a 1.

La línea 750 altera los registros del PSG 6 y 8, logrando así el sonido de los reactores. En la 760 se hace desaparecer de la pantalla el último horizonte dibujado y en las líneas 770 y 780 se calcula la inclinación del avión respecto del horizonte. Desde las líneas 790 hasta la 820 se controla el movimiento del horizonte, impidiendo que éste salga fuera de los límites de la supuesta ventana del avión.

La línea 830 dibuja por fin el nuevo horizonte calculado. Entre las líneas 840 a 920 se realizan llamadas a varias subrutinas encargadas de mostrar en pantalla la causa del accidente.

Las líneas comprendidas entre la 940 y la 970 se encargan de mostrar durante algún tiempo la pista de aterrizaje elegida por la torre de control.



LA APROXIMACION FINAL

- RUTINA DE CONTROL DEL TECLADO
- COMPROBACION OE ATERRIZAJE SOBRE LA PISTA
- CHOQUES
- ESTIMACION OE OAÑOS

Quita el piloto automático, pero no exhales todavía el suspiro de alivio: aún tienes los mandos. Utilizando únicamente seis teclas debes conseguir que el avión se pose suavemente sobre la tierra.

Hay once teclas para controlar el avión, que nos permiten aumentar o disminuir la velocidad, hacer que el avión vire o que suba y baje. Además tú mismo podrás elegir la altura y la distancia, respecto a la pista de aterrizaje, a la que quieres empezar.

No es fácil conducir un avión, en especial si eres un piloto inexperto. No esperes convertirte en experto con unos cuantos vuelos: la capacidad de pilotar un avión es una habilidad que muy pocos adquieren en poco tiempo y con facilidad.

Cuando vayas perseverando y adquiriendo más experiencia en el control de la aeronave, tus maniobras de aterrizaje irán mejorando.

Si no logras aterrizar con éxito el ordenador mostrará en la pantalla el error cometido. Estúdialo para ver lo que has hecho mal, y tenlo en cuenta la próxima vez.

```
490 K$=INKEY$
500 IF K$="Q" THEN A1=
    A1-1
510 IF K$="A" THEN A1=
    A1+1
520 IF K$="W" THEN
    A1=A1-100
530 IF K$="S" THEN
    A1=A1+100
540 IF K$="I" THEN
    V1=V1+10
550 IF K$="K" THEN
    V1=V1-10
560 IF K$="U" THEN
```



```
      V1=V1+100
570  IF K$="J" THEN
      V1=V1-100
580  IF K$="O" THEN
      G1=G1-1
590  IF K$="P" THEN
      G1=G1+1
600  IF K$="O" THEN
      Y1=Y1-1:Y2=
      Y2+1
610  IF K$="P" THEN
      Y1=Y1+1:Y2=
      Y2-1
980  SCREEN 0
990  PRINT "TE HAS ELEVADO
      DEMASIADO"
1000 PRINT "EL FUSELAJE NO
      HA AGUANTADO"
1010 PRINT "QUE TE CREES
      QUE TU TRASTO ES
      UN "
1020 PRINT "CONCORDE."
1030 GOTO 1180
1040 SCREEN 0
1050 PRINT "HAS CAIDO A
      TIERRA, DONDE TE
      DIERON EL CARNET, EN
      UNA TOMBOLA?"
1060 PRINT "POR TU CULPA
      HAN MUERTO TODOS LOS
      PASAJEROS."
1070 GOTO 1180
1080 SCREEN 0
1090 PRINT "TE HAS QUEDADO
      SIN COMBUSTIBLE."
1100 PRINT "GASTAS MAS QUE
      LA VESPA DE MI
      PRIMO."
1110 GOTO 1180
1120 SCREEN 0
1130 PRINT "DEBIDO A LA
      POCA VELOCIDAD SE TE
      HAN PARADO LOS
      MOTORES Y HAS CAIDO
```



```
      EN PICADO."
1140 GOTO 1180
1150 SCREEN 0
1160 PRINT "LA VELOCIDAD HA
      SIDO EXCESIVA. HAS
      PARTIDO LAS
      ALAS."
1170 GOTO 1180
1180 PRINT:PRINT:PRINT:SOUND
      6,0:SOUND 8,0
1190 PRINT "QUIERES VOLAR
      DE NUEVO.
      (S/N)"
1200 INPUT SN$
1210 IF SN$="S" OR SN$="s"
      THEN RUN
1220 END
1330 SCREEN 0
1340 PRINT "HAS CONSEGUIDO
      ATERRIZAR ERES UN
      HEROE, NO CABE
      DUDA."
1350 PLAY "O8CDEFGABO7BAG
      FEDCO6CDEFGABO5BAGF
      EDC"
1360 GOTO 1180
1370 SCREEN 0
1380 PRINT "HAS TOMADO
      TIERRA A DEMASIADA
      VELOCIDAD"
1390 PRINT "HAS QUEMADO
      LOS NEUMATICOS."
1400 GOTO 1180
1410 SCREEN 0
1420 PRINT "HAS PARADO LOS
      MOTORES ANTES DE
      TOMAR
      TIERRA."
1430 PRINT "HAS CAIDO COMO
      UNA PIEDRA SOBRE LA
      PISTA."
1440 GOTO 1180
1450 SCREEN 0
1460 PRINT "SE TE HA
      ACABADO LA PISTA"
1470 PRINT "HAS CHOCADO
      CONTRA LA TORRE DE
      CONTROL."
1480 GOTO 1180
1490 SCREEN 0
1500 PRINT "HAS TOMADO
      TIERRA CON UN
      DESCENSO DEMASIADO
      ACELERADO"
```



```
1510 GOTO 1180
1520 SCREEN 0
1530 PRINT "NO HAS TOMADO
     TIERRA SOBRE LA
     PISTA."
1540 GOTO 1180
1550 SCREEN 0
1560 PRINT "HAS VIRADO EL
     AVION SOBRE LA CIUDAD
     Y HAS CHOCADO CONTRA
     UN RASCACIELOS."
1570 GOTO 1180
```

Entre las líneas 496 y 610 se encuentra la rutina que nos permite gobernar el avión. A partir de la línea 980 y hasta la 1330 aparecen un conjunto de subrutinas donde se almacena la información referente a la causa de los accidentes más frecuentes. La primera subrutina va de la línea 980 hasta la 1030, y aparecerá en tu monitor cuando la altura de vuelo haya sido excesiva. La segunda subrutina, líneas 1040 a 1070, aparecerá cuando te hayas precipitado contra el suelo. La siguiente subrutina se encuentra entre las líneas 1080 y 1110 y te será mostrada para informarte de que te has quedado sin gota de combustible. La cuarta subrutina, 1120 a 1140, contiene el mensaje de accidente por paro de los motores, y éste es causado por una velocidad demasiado baja. Si por el contrario la velocidad ha sido excesiva corres el riesgo de partir las alas del avión y caer en picado. Si te sucediera esto aparecería en pantalla el mensaje de las líneas 1150, 1160 y 1170.

La rutina situada entre las líneas 1180 y 1220 es empleada para terminar el juego, y gracias a ella podremos escoger entre reinicializar la partida o acabar y volver al BASIC.

ACCIDENTES AL LLEGAR AL AEROPUERTO

Las subrutinas contenidas entre las líneas y el final del listado son las que guardan la información acerca de los accidentes sobre el aeropuerto. La primera de ellas va desde la línea 1330 a la 1360 y guarda la información para los casos en que se logra aterrizar. La segunda subrutina de este bloque va de la línea 1370 a la 1400 y es para los casos de accidente debidos a un exceso de velocidad. La siguiente subrutina situada entre las líneas 1410 y 1440 se visualizará en pantalla en caso de parar los motores antes de tomar tierra; si por el contrario éstos son parados demasiado tarde es muy probable que la pista de aterrizaje se te quede corta. En este caso aparecerá el mensaje contenido entre las líneas 1450 y 1480. Entre las líneas 1490 y 1510 se encuentra la subrutina que te informará en caso de accidente por haber realizado un descenso demasiado acelerado. La penúltima subrutina hará aparecer el texto en pantalla en el caso de que no hayas logrado alinear correctamente el avión respecto a la pista de aterrizaje.

Y la última subrutina controla el accidente provocado por virar sobre una gran ciudad.



LAS SERPIENTES SUMADORAS

- EL CLASICO JUEGO EN UNA VERSION PARA BASIC
- DIBUJO DE LA COMIDA
- COMIENDO NUMEROS
- LA CULEBRA SE EXTIENDE

Guía a la serpiente hambrienta para que coma en el «Juego de la serpiente de INPUT». Al engullir ávidamente los suculentos números, la pequeña serpiente irá creciendo hasta hacerse enorme. Naturalmente, siempre que tengas la necesaria destreza.

El juego de la serpiente es uno de los más conocidos y más fáciles de jugar, pese a lo cual continúa siendo enormemente enviciante. Por suerte, no hace falta recurrir al código máquina para programar un juego de este tipo; precisamente este es un juego que ha marcado época en la historia de los ordenadores domésticos, figurando como uno de los más satisfactorios que pueden escribirse en BASIC.

JUGANDO EL JUEGO

El objeto del juego es ir guiando a la hambrienta serpiente por la pantalla, de forma que vaya engullendo los números que van apareciendo de forma aleatoria. Cada vez que la serpiente se traga un número, su longitud aumenta en el correspondiente número de segmentos.

Ten cuidado de no sobrepasar los bordes y de no permitir que la serpiente se entrecruce consigo misma, cosa que te irá resultando más difícil a medida que se va haciendo más larga. Si atraviesas uno de los bordes o el cuerpo de la serpiente, el juego terminará, mostrándote tu error junto con la clásica opción de volver a jugar.

COMENTARIO

En la línea 10 abrimos un fichero que nos permite escribir caracteres en una pantalla de gráficos; desde la línea 50 a la línea 90 se dibuja el aspecto de la pantalla; la línea 100 sitúa la posición de nuestro personaje en la pantalla, mientras que la siguiente línea define el máximo número de movimientos que podemos realizar.

Por su parte, la línea 120 pone la variable TIME a cero; esta variable será utilizada más tarde para ir hacer apareciendo progresivamente números aleatorios cada seis segundos, aproximadamente.

Desde la línea 130 a la línea 240 se lee el teclado con el fin de desplazar la serpiente por la pantalla.

A partir de la línea 350 hasta la 370 el ordenador comprueba si la serpiente se sale de los márgenes. La línea 380 registra el último movimiento que hemos realizado.

A continuación, las líneas comprendidas entre 380 y 410 se encargan de ir borrando la última parte de la cola conforme avanzamos. A su vez, la línea 430 se encarga de imprimir otro número aleatorio en caso de que hayan pasado seis segundos; para ello hace proseguir el control hacia la línea 440; en caso de que no hayan pasado esos seis segundos el control pasa a la línea 550.

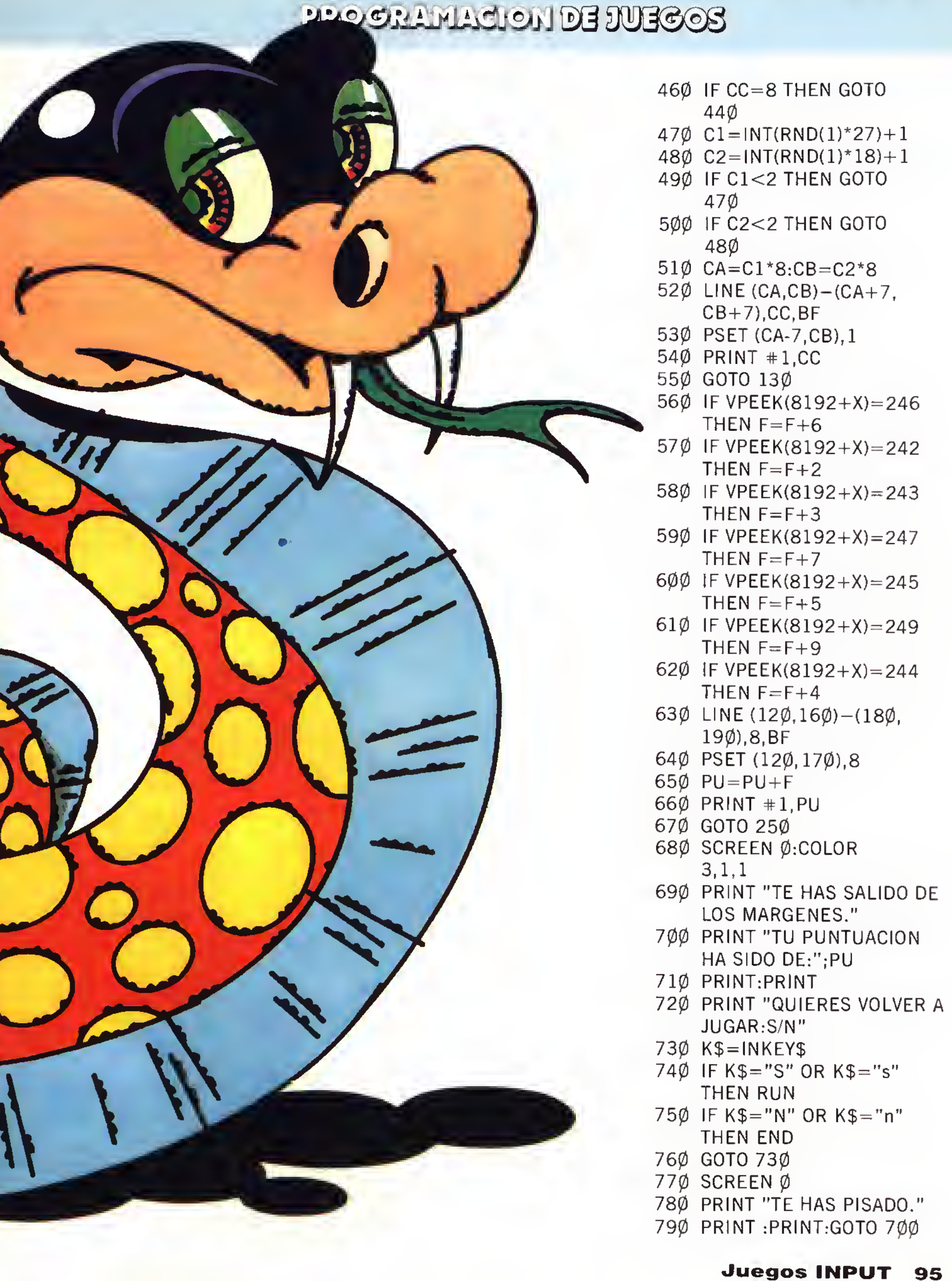
En seguida, la línea 560 empieza la rutina empleada para incrementar la puntuación y longitud de nuestra serpiente.

A partir de la línea 680 se encuentra la rutina llamada en caso de que la serpiente se salga de los bordes y, por último, en la línea 770 empieza la rutina seleccionada en caso de que el reptil se cruce consigo mismo.



La serpiente se desplaza guiada por las teclas de cursor.

```
10 OPEN "GRP:" AS #1
20 F=1
30 Z=1
40 COLOR 15,1,15
50 SCREEN 2
60 LINE (10,10)-(245,160),
   8,B
70 PAINT (1,1),8
80 PSET (40,170),1
90 PRINT #1,"PUNTUACION:"
100 X=528
110 DIM A(1600)
120 TIME=0
130 D=STICK(0)
140 PA=X
150 IF D=1 THEN X=X-
    256
160 IF D=3 THEN X=X+8
170 IF D=5 THEN X=X+
    256
180 IF D=7 THEN X=X-8
190 IF INKEY$=" " THEN
    F=F+5
200 IF VPEEK(X+8192)<>1
    THEN GOTO 560
210 IF VPEEK(PA+8)=60 AND
    D=3 THEN 770
220 IF VPEEK(PA-8)=60 AND
    D=7 THEN 770
230 IF VPEEK(PA+256)=60
    AND D=5 THEN 770
240 IF VPEEK(PA-256)=60
    AND D=1 THEN 770
250 VPOKE X+0,&B00111100
260 VPOKE X+1,&B01111110
270 VPOKE X+2,&B11111111
280 VPOKE X+3,&B11111111
290 VPOKE X+4,&B11111111
300 VPOKE X+5,&B01111110
310 VPOKE X+6,&B00111100
320 VPOKE X+7,&B00000000
340 IF D=0 THEN GOTO
    430
350 IF X=>4864 THEN GOTO
    680
360 IF X<0 THEN GOTO
    680
370 IF VPEEK(8192+X)=8
    THEN 680
380 A(Z)=X
390 FOR T=A(Z-F) TO
    A(Z-F)+7
400 VPOKE T,0
410 NEXT T
420 Z=Z+1
430 IF TIME>300 THEN
    TIME=0 ELSE GOTO 550
440 CC=INT(RND(1)*9)+1
450 IF CC=1 THEN GOTO 440
```



```
460 IF CC=8 THEN GOTO
    440
470 C1=INT(RND(1)*27)+1
480 C2=INT(RND(1)*18)+1
490 IF C1<2 THEN GOTO
    470
500 IF C2<2 THEN GOTO
    480
510 CA=C1*8:CB=C2*8
520 LINE (CA,CB)-(CA+7,
    CB+7),CC,BF
530 PSET (CA-7,CB),1
540 PRINT #1,CC
550 GOTO 130
560 IF VPEEK(8192+X)=246
    THEN F=F+6
570 IF VPEEK(8192+X)=242
    THEN F=F+2
580 IF VPEEK(8192+X)=243
    THEN F=F+3
590 IF VPEEK(8192+X)=247
    THEN F=F+7
600 IF VPEEK(8192+X)=245
    THEN F=F+5
610 IF VPEEK(8192+X)=249
    THEN F=F+9
620 IF VPEEK(8192+X)=244
    THEN F=F+4
630 LINE (120,160)-(180,
    190),8,BF
640 PSET (120,170),8
650 PU=PU+F
660 PRINT #1,PU
670 GOTO 250
680 SCREEN 0:COLOR
    3,1,1
690 PRINT "TE HAS SALIDO DE
    LOS MARGENES."
700 PRINT "TU PUNTUACION
    HA SIDO DE:";PU
710 PRINT:PRINT
720 PRINT "QUIERES VOLVER A
    JUGAR:S/N"
730 K$=INKEY$
740 IF K$="S" OR K$="s"
    THEN RUN
750 IF K$="N" OR K$="n"
    THEN END
760 GOTO 730
770 SCREEN 0
780 PRINT "TE HAS PISADO."
790 PRINT :PRINT:GOTO 700
```

ADIVINA MIS PALABRAS

- UN JUEGO PARA DOS JUGADORES
- PREPARACION DE LA PANTALLA
- LAS REGLAS DEL JUEGO
- VALORES DE LAS LETRAS
- ESTRATEGIA

Introduce una buena palabra para los juegos de ordenador docentes. El juego de las palabras de INPUT es adecuado para todas las edades, puede hacerse tan difícil o tan fácil como desees, y es increíblemente adictivo.

Los juegos de ordenador no tienen por qué ser únicamente recreativos como los juegos para salón o como algunos de simulación domésticos, sino que también pueden ser docentes.

«El Ahorcado» es el conocido juego que puede convertirse para que corra en un ordenador. Este juego puede ayudar a la gente en su expresión, en sus conocimientos generales, etc. Elige un tema como el de la ingeniería química y pronto dominarás una gran parte de su terminología.

El juego de palabras de INPUT es de este estilo, es para dos jugadores y se basa en adivinar palabras o frases. Es más interesante y divertido que «El Ahorcado», y es igual o más educativo.

Puedes jugarlo como «El Ahorcado», limitado a un determinado tema, puedes utilizar palabras con un determinado número de letras, emplear citas de Cervantes o lo que mejor te parezca.





EL JUEGO

Introduce los nombres de los dos jugadores. Después tienes la opción de elegir el número de letras de la frase que introduce un jugador. Una interesante faceta de este juego es que cuanto más largas son las frases, algunas veces son más fáciles de adivinar porque mayor es el número de pistas. Prueba y verás.

A continuación, el primer jugador deberá pensar en una frase e introducirla. No es necesario que el oponente esté encerrado y gritando en un armario mientras se efectúa la introducción.

Entre cada palabra de una frase sólo debe haber un espacio. La longitud máxima de cualquier frase es de 30 caracteres en el MSX.

Una vez completada la frase, hay que pulsar la tecla de entrada y aparece la pantalla principal. En la parte inferior de la misma hay la puntuación del jugador. Al principio del juego, el jugador tiene 200 puntos, y el total puede aumentar o disminuir a medida que progresa el juego. Encima de las puntuaciones hay una tabla con los valores de las letras: las más corrientes tienen valores altos, y las menos corrientes, valores bajos. La frase misteriosa se presenta en forma de una fila de rayas.

En el fondo de la pantalla aparece un juego de instrucciones y un espacio para que introduzcas tus comandos y tus adivinanzas.

ESTRATEGIA

El jugador tiene tres opciones: comprar letras, adivinar una letra en una posición específica, o adivinar la frase.

En las primeras etapas del juego, una buena elección es comprar un espacio, aunque hay que asegurarse de que la frase contiene más de una palabra. La forma de proceder a partir de ahí es asunto tuyo. Las vocales son caras, pero tienen una probabilidad muy elevada de que sean válidas, mientras que las letras más baratas son arriesgadas debido a su rareza. A veces, las palabras son más fáciles de adivinar si se han encontrado algunas consonantes —un derroche de vocales no siempre es de ayuda.

A medida que la frase va tomando forma, probablemente verás que puedes adivinar una letra de una determinada posición. Por ejemplo, puedes tener una palabra como Q.E . Es evidente que la letra central es una U. De esta manera puedes sumar puntos. Una letra elegida correctamente sumará su valor a tu puntuación, mientras que una errónea sólo restará la mitad de su valor. Teclea 2 para seleccionar la opción de adivinanza e introduce la tuya.

Cuando haya varias letras en su lugar, es posible que tengas un destello de inspiración y desees adivinar toda la frase. Para ello, teclea 3 y la podrás introducir.

Si la frase es la correcta el valor de tu puntuación actual se triplicará. En cambio, si es errónea, se restarán 50 puntos. Un exceso de adivinanzas a tontas y a locas pronto dejará a cero tu puntuación.

Ahora introduce la primera sección del juego de las palabras. Estas líneas dejan la pantalla preparada para empezar nuestro juego de las palabras.

Si haces RUN verás que el programa no funciona correctamente puesto que el resto del programa (que cubre las distintas elecciones) se termina en el siguiente capítulo.

No olvides salvar esta parte del programa con un buen SAVE.

```
10 CLS
20 INPUT "NOMBRE DEL
   JUGADOR A:";Z1$
30 INPUT "NOMBRE DEL
   JUGADOR B:";Z2$
40 CLS
50 PU=200
60 PRINT Z1$
70 INPUT "CUANTOS
   CARACTERES,INCLUIDOS
   ESPACIOS:";E
80 IF E>30 THEN GOTO 70
90 INPUT "CADENA:";C$
100 DIM SM$(E), SM(E), L(E:,
    F1$(E)
110 FOR T=1 TO E
120 DM$=MID$(C$,T,1)
130 SM$(T)=DM$: SM(T)=T+5
140 NEXT T
150 FOR T=1 TO E
160 F1$(T)=" "
```



```
170 NEXT T
180 FOR T=1 TO E
190 FV$=FV$+F1$(T)
200 NEXT T
210 CLS
220 LOCATE 6,10
230 PRINT FV$
240 LOCATE 10,1
250 PRINT Z2$
260 GOSUB 720
270 LOCATE 20,22
280 PRINT "PUNTUACION:";
    PU
290 LOCATE 0,18
300 IF PU=<0 THEN GOTO
    1140
310 PRINT "1-COMPRAR"
320 PRINT "2-INTRODUCIR
    CARACTER"
330 PRINT "3-INTRODUCIR
    FRASE"
340 K$=INKEY$
350 IF K$="1" THEN GOTO
    390
360 IF K$="2" THEN GOTO
    830
370 IF K$="3" THEN GOTO
    980
380 GOTO 340
390 INPUT "CARACTER:";
    CA$
400 IF CA$="" THEN CA$=" "
410 FOR T=1 TO E
420 IF CA$=SM$(T) THEN
    L(T)=1:CP=CP+1
430 NEXT T
```

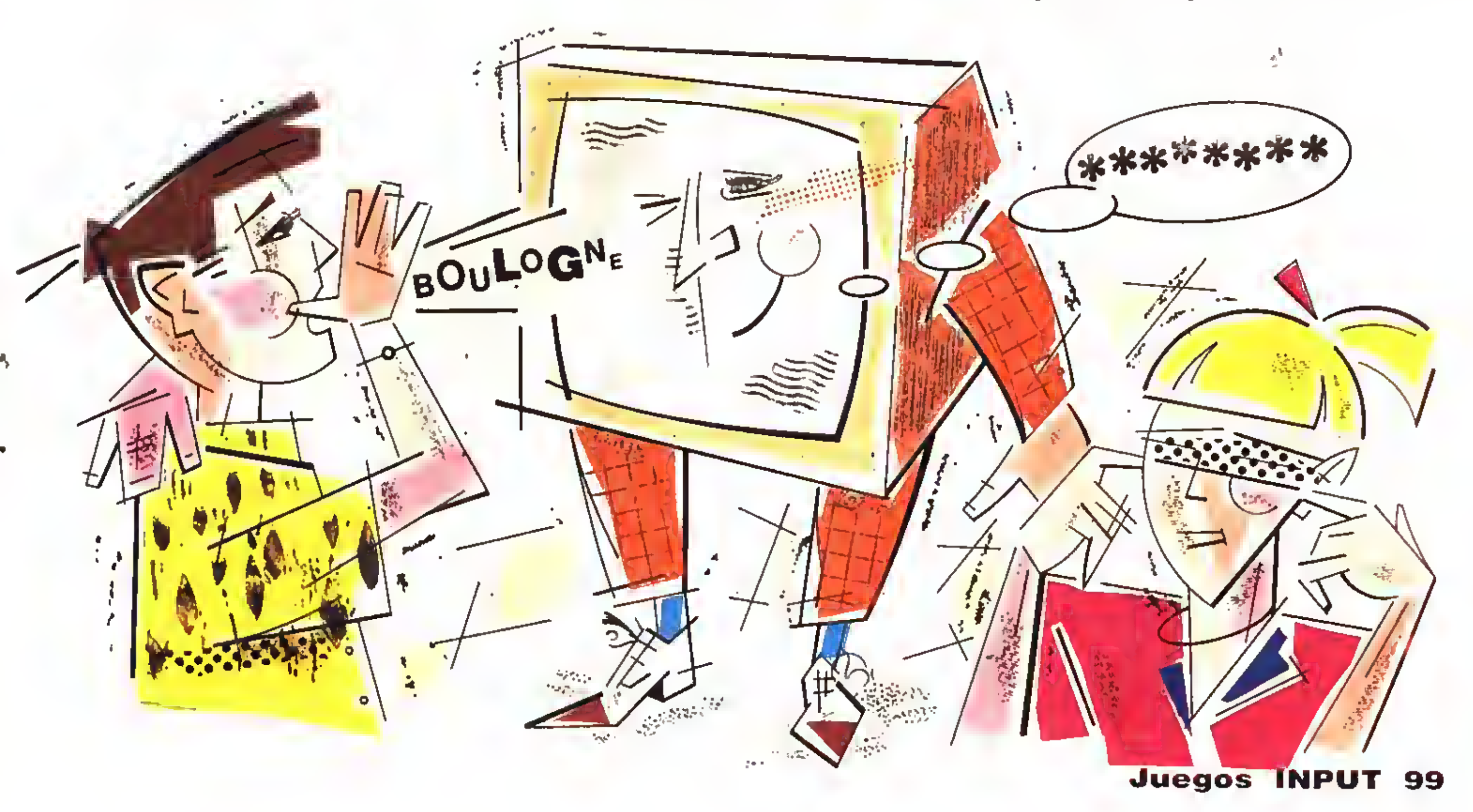
```
440 FOR T=1 TO E
450 IF L(T)<>0 THEN GOTO
    480
460 NEXT T
470 GOSUB 640
480 FV$=""
490 FOR T=1 TO E
500 IF L(T)=0 THEN
    FV$=FV$+"-"
510 IF L(T)<>0 THEN
    FV$=FV$+SM$(T)
520 NEXT T
530 RESTORE 820
540 FOR T=65 TO 90
550 READ Q
560 IF CA$=CHR$(T) THEN
    PU=PU+Q*CP
570 NEXT T
580 CP=0
590 FOR T=1 TO E
600 IF MID$(FV$,T,1)="-" THEN
    GOTO 630
610 NEXT T
620 GOTO 1050
630 GOTO 210
640 CLS:PRINT "EL CARACTER
    ";CA$;" NO SE ENCUENTRA
    EN LA FRASE"
650 FOR T=1 TO 1000:
    NEXT T
```

El programa es muy sencillo, puesto que no hay gráficos que requieran el empleo de funciones especiales para visualizarlos.

La línea 10 borra la pantalla y las 20-30 preguntan el nombre de los dos jugadores, es decir el que más tarde introducirá la frase y el que tendrá que adivinarla. La línea 50 inicializa nuestra puntuación con 200 puntos. En líneas 60-70 informa al jugador A que introduzca el número de caracteres ocupados por la frase misteriosa. La línea 80 comprueba que este número no exceda de 30, la 90 pide al usuario que introduzca la cadena de caracteres a adivinar.

Por su parte, la línea 100 dimensiona cuatro variables.

El bucle situado entre 110 y 140 se encarga de introducir cada carácter de la cadena en una variable de matriz. Los dos bucles siguientes, situados entre 150 y 200, se encargan de crear una variable formada con tantas rayas como caracteres tiene la cadena introducida anteriormente. La línea 210 borra de nuevo la pantalla, 220 y 230 imprimen en pantalla, la línea 240 y siguiente imprime el nombre del segundo jugador. La línea 260 llama a una subrutina que será explicada en el próximo capítulo. 270 y 280 imprimen la puntuación en pantalla, la línea 300 comprueba si la puntuación ha llegado a cero. Desde las líneas 310 a 330 se muestra un menú de opciones para el jugador, desde la línea 340 a 380 el ordenador esperará a que el jugador escoja una opción. En la línea 390 empieza la rutina de compra de caracteres, la línea 400 comprueba si el dato introducido es un espacio, el bucle situado entre las líneas 410 y 430 comprueba si nuestro carácter está contenido en la cadena a adivinar, el siguiente bucle, entre las líneas 440 y 460 se cercioran de que el bucle anterior haya encontrado algún carácter igual a los buscados para seguir con la rutina. La línea 470 lanza una llamada a la rutina situada en la línea 640, esta rutina nos informa de que el carácter buscado no se encuentra en la frase misteriosa. Desde la línea 480 a la línea 520 el ordenador construye una nueva frase a buscar sustituyendo las rayas por las letras adivinadas. Desde las líneas 540 a la 570 suma la puntuación apropiada por cada letra adivinada, desde la línea 590 a 610 se comprueba si aún queda alguna letra por adivinar y por último la línea 630 devuelve el control a la línea 210, donde el proceso se repite de nuevo.



ULTIMANDO EL JUEGO DE LAS PALABRAS

Saca quinario, xileno y quincuagésimo de los polvorientos rincones de tu diccionario.

Puedes utilizar tácticas «sucias» para sacar de quicio a tus amigos con el juego de las palabras.

En la primera parte de nuestro juego *Adivina mis palabras* habíamos introducido los nombres de los dos jugadores y, a continuación, habíamos elegido el número de letras de la frase a entrar.

En seguida, el primer jugador había pensado en una frase y la había introducido. Una vez completada la frase, se había pulsado la tecla de entrada y había aparecido la pantalla principal.

Recordemos que el jugador tenía tres opciones: comprar letras, adivinar una letra en una posición específica, o adivinar la frase.

Decíamos, además, que en las primeras etapas del juego era una acertada medida comprar un espacio, aunque había que asegurarse bien de que la frase contuviera más de una palabra.

Al llegar a este punto, la forma de proceder quedaba por completo a tu arbitrio.

En seguida, habías introducido la primera sección del programa, y estas líneas dejaban la pantalla a punto para empezar el juego de las palabras.

COMIENZA EL JUEGO

Ahora, una vez recapitulada la fase preparatoria, estamos en condiciones de introducirnos de lleno en este atractivo pasatiempo informático.

En esta segunda parte final sobre el juego de las palabras de INPUT hay todo lo que necesitas para empezar a jugar.

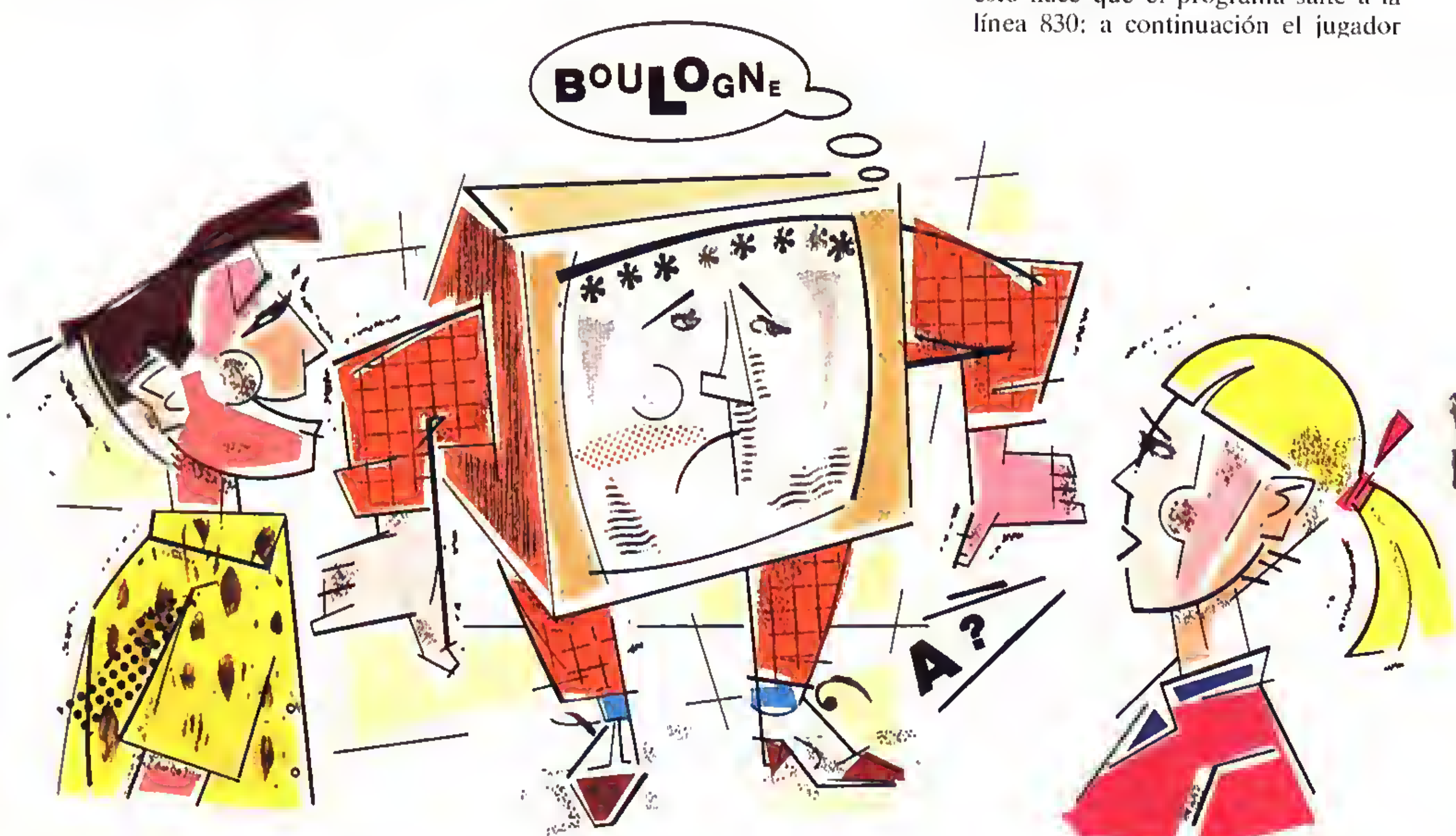
Teclea ahora el resto de líneas para ver algunas interesantes aplicaciones del manejo de cadenas con tu máquina.

Después trata de desconcertar a tus amigos con oscuras palabras, y si un jugador decide comprar una letra, lo primero que hace el ordenador es comprobar su valor. Para ello el programa se dirige a la rutina que empieza en la línea 540.

Ésta comprueba el valor del carácter basándose en una tabla de valores situada en los data del programa: esta misma rutina actualiza la puntuación del jugador.

LETRAS ESPECIFICAS

Si el jugador desea adivinar una letra específica, deberá seleccionar dos; esto hace que el programa salte a la línea 830: a continuación el jugador



- JUEGO DE LAS PALABRAS COMPLETO CON NUEVAS RUTINAS
- COMPRA DE LETRAS
- COMPROBACION DE LA ENTRADA DEL JUGADOR
- COLOCACION DE UNA LETRA ESPECIFICA EN POSICION
- ACERTAR LA FRASE COMPLETA
- ¡TRAMPA!

deberá introducir el carácter elegido, y después de coma el lugar que ocupa dentro de la frase. Si la letra es errónea, el ordenador imprimirá MALA SUERTE; de lo contrario, el mensaje será FELICIDADES. Si con esto la frase ha quedado completa, el ordenador imprimirá además HAS ADIVINADO LA FRASE.

LA FRASE ENTERA

Si el jugador que adivina es más ambicioso y ha elegido la opción de adivinar toda la frase (3), la línea 370 envía el programa a la línea 980. La rutina pide al jugador que introduzca la frase, la cual es comparada con la que se ha introducido inicialmente. Si la frase es errónea, se restan 50 puntos de la puntuación del jugador. Si es correcta, la puntuación actual del jugador es triplicada, terminando el juego con el mensaje HAS ADIVINADO LA FRASE mostrándote tu puntuación final.

EL FINAL

Si el juego ha terminado, el programa salta a la línea 1050. En ella y siguientes se muestra la puntuación. El programa termina aqui, aunque el ordenador te ofrece la posibilidad de jugar de nuevo en la rutina situada entre las líneas 1080 y 1130.

```
66Ø  RESTORE 82Ø
67Ø  FOR T=65 TO 9Ø
68Ø  READ Q
69Ø  IF CA$=CHR$(T) THEN
     PU=PU-Q/2
7ØØ  NEXT T
71Ø  RETURN
72Ø  LO=2:L2=2: CR=65:
     RESTORE 82Ø
73Ø  LOCATE LO,L2
74Ø  READ Q
75Ø  PRINT CHR$(CR);"=
     ";Q
76Ø  LO=LO+1Ø
77Ø  IF LO=>35 THEN
     LO=2:L2=L2+1
78Ø  CR=CR+1
79Ø  IF CR=91 THEN GOTO
     81Ø
8ØØ  GOTO 73Ø
81Ø  RETURN
82Ø  DATA 2Ø,1Ø,1Ø,12,2Ø,Ø8,
     12,Ø8,2Ø,Ø4,Ø6,1Ø,1Ø,1Ø,
     2Ø,1Ø,Ø2,12,12,12,2Ø,Ø8,
     Ø8,Ø4,Ø8,Ø2
83Ø  ' RUTINA DE
     INTRODUCCION DE
     CARACTER
84Ø  INPUT "CARACTER,
```



```
     POSICION:";CA$,
     PO
850  CLS
860  FOR T=1 TO E
870  IF T=PO AND SM$(T)=CA$
     THEN AC=1:GOTO 890
     ELSE AC=0
880  NEXT T
890  IF AC=1 THEN GOTO
     960
900  RESTORE 820:PRINT
     "MALA SUERTE"
910  FOR T=65 TO 90
920  READ Q
930  IF CA$=CHR$(T) THEN
     PU=PU/2
940  NEXT T
950  GOTO 210
960  RESTORE 820:PRINT
     "FELICIDADES"
970  GOTO 410
980  INPUT "FRASE:";
     FA$
990  IF FA$=C$ THEN PRINT
     "FELICIDADES":GOTO
     1030
1000  PRINT "MALA SUERTE:"
1010  PU=PU-50
1020  GOTO 210
1030  PU=PU*3
1040  GOTO 1050
1050  CLS
1060  PRINT "HAS ADIVINADO
      LA FRASE:"
1070  PRINT
      "TU PUNTUACION
      HA SIDO DE:";PU
1080  PRINT
1090  PRINT "QUIERES VOLVER
      A JUGAR"
1100  K$=INKEY$
1110  IF K$="S" OR K$="s"
      THEN RUN
1120  IF K$<>"" AND K$<>"S"
      AND K$<>"s" THEN
      END
1130  GOTO 1100
1140  CLS
1150  PRINT "TU PUNTUACION
      ES CERO, NO LO HAS
      CONSEGUIDO."
1160  PLAY
      "O2V15L408AGFEDC"
1170  GOTO 1080
```



BUENO COMO EL ORO

- UN JUEGO OE ESTRATEGIA EN LOS NEGOCIOS
- OESCRIPCION OEL JUEGO
- INGRESOS Y GASTOS
- LA RUTINA DE ROBO

Vive los riesgos y las satisfacciones de los grandes negocios con el juego de la mina de oro de INPUT. ¿Estás seguro de poseer la habilidad y el buen juicio necesarios para adoptar las decisiones correctas y actuar consecuentemente con ellas?

La **Mina de Oro** es un juego de estrategia comercial en el que tú haces el papel del propietario de una empresa minera. Tu misión es hacer que la compañía prospere lo mejor posible. Durante el desarrollo del juego se te irán presentando constantemente una serie de opciones entre las que tienes que elegir, y los destinos de la compañía dependen de tu habilidad para tomar decisiones prudentes e imaginativas.

Al igual que los juegos de aventuras, los de estrategia en los negocios generalmente se escriben íntegramente en BASIC, no siendo necesaria la alta velocidad del código máquina. Además, debido a que no requieren grandes secciones de texto, es relativamente sencillo escribirlos en versiones para ordenadores que no tengan mucha memoria. En este caso hemos adornado el juego de la **Mina de Oro** añadiéndole unos gráficos que muestren el progreso de la prospección minera, con lo cual aumentan bastante las necesidades de memoria, si bien todavía te cabe el programa en un **MSX** de 16 K.

El programa es con todo relativamente largo, por lo que lo hemos dividido en dos partes. En este artículo veremos la forma de establecer el núcleo del juego y en el próximo veremos algunas de las rutinas que necesitas para que el juego se pueda realmente jugar.

Cuando hayas introducido todo el listado que figura más adelante, haz un SAVE para guardarlo hasta la próxima vez.

DESCRIPCION DEL JUEGO

Al empezar el juego, tus dos posesiones son la empresa minera y 2 millones de dólares en efectivo. Tienes que invertirlos inteligentemente en la exploración del metal precioso. El objetivo del juego es ganar tanto dinero como sea posible en 30 pasadas. Puedes jugar tú solo o contra un oponente que tenga el control de una compañía rival.

En cada pasada se te presentan unas cuantas alternativas entre las que elegir. Antes de empezar con las labores de minería, debes encontrar un emplazamiento adecuado, por lo que has de invertir para tener un informe de prospecciones. Con el informe anterior, podrás estimar tus posibilidades de encontrar oro, su profundidad probable y la cantidad esperada. Tienes que decidir si merece la pena o no iniciar la explotación.

La minería es muy cara, por lo que podrías decidir inversiones en investigación y desarrollo de nuevos equipos que te permitan reducir tus costes. O tal vez es mejor iniciar directamente las excavaciones, sólo tú puedes decidir.

Si das comienzo a las excavaciones, hay una representación gráfica que te permitirá apreciar el progreso de la mina. Si no se encuentra oro, puedes elegir entre continuar excavando o abandonar la mina e iniciar un nuevo trabajo.

A lo largo del desarrollo, hay otros dos factores que entrarán en juego. Cuando encuentres oro, puedes elegir entre almacenarlo en tus cámaras acorazadas o venderlo en el mercado de metales preciosos. Puede que sea razonable conservarlo, en el supuesto de que no necesites dinero urgentemente, esperando hasta que alcance un precio más ventajoso; el precio del oro está fluctuando durante todo el juego. Pero ten cuidado, porque hay ladrones de oro y cuanto mayor cantidad tengas almacenada, más tentador es su premio.

En la segunda parte de este artículo veremos con mayor profundidad los trabajos asociados con el juego. Pero ahora teclea ya la primera parte del programa.

```
5 COLOR 3,1,1:CLS
1Ø LOCATE 2,
   9:PRINT"CUANTOS
   JUGADORES? (1 o
   2)":A$=INKEY$:IF A$=""
   THEN GOTO 1Ø
2Ø IF A$<"1" OR A$>"2" THEN
   GOTO 1Ø
3Ø IF A$="1" THEN P=1 ELSE
   P=2
4Ø DIM A(2,6):DIM C(2,5):DIM
   A$(P):DIM R(2):ER=1ØØØØ
5Ø R(1)=Ø:R(2)=Ø:A(1,1)=2Ø
   ØØØØ#:A(1,2)=2ØØØØØØ
   #:A(2,1)=2ØØØØØØ#:A(2,2
   )=2ØØØØØØ#:A(1,3)=Ø:A(2
   ,3)=Ø:A(1,4)=1ØØØØØ!:A(2
   ,4)=1ØØØØØ!:A(1,5)=Ø#:A
   (2,5)=Ø:A(1,6)=Ø:A(2,6)=
   Ø:PRINT
7Ø FOR N=1 TO P:PRINT
   "NOMBRE DEL JUGADOR ";
   N;:INPUT A$(N):NEXT N
2ØØ FOR N=1 TO 3Ø:FOR M=1
    TO NOP
2Ø2 COLOR 15,1,1:CLS
21Ø LOCATE 4,Ø:PRINT " M I N
    A  D E  O R O "
22Ø PRINT TAB(16);A$(1) :IF
    NOP=2 THEN PRINT
    TAB(24);A$(2)
23Ø PRINT "ACTIVOS TOT.$";
    TAB(15); A(1,1) : IF NOP=2
    THEN PRINT TAB(24);A(2,
    1);
```

```
240 PRINT "CAJA EFECTIVO $";
    TAB(15);A(1,2) :IF NOP=2
    THEN PRINT TAB(24);A(2,
    2);
250 PRINT "CANTIDAD DE ORO
    KG";TAB(15);A(1,3) :IF
    NOP=2 THEN PRINT
    TAB(24);A(2,3);
260 PRINT "COSTES MINER.$";
    TAB(15);A(1,4):IF NOP=2
    THEN PRINT TAB(24);A(2,
    4);
270 PRINT "NUM DE MINAS";
    TAB(15); A(1,5):IF NOP=2
    THEN PRINT TAB(24);A(2,
    5);
280 PRINT "PROFUND.MIN M";
    TAB (15);A(1,6):IF NOP=2
    THEN PRINT TAB(24);A(2,
    6);
300 PRINT "PRECIO DEL
    ORO:—":PRINT "$";ER; "
    POR KG DE ORO"
400 PRINT ">—";A$(M)
500 PRINT "1";:PRINT
    "—INVESTIGACION Y
    DESARROLLO"
510 PRINT "2";:PRINT
    "—EXPLORACION E
    INFORME"
520 PRINT "3";:PRINT "—INCR.
    PROFUN. MINA EN 200
    METROS"
530 PRINT "4";:PRINT
    "—PRECIO DEL ORO EN
    DOLARES"
```



```
54Ø PRINT "5";:PRINT
    "-PASO"
55Ø PRINT:PRINT "TECLEA UNA
    INSTRUCCION"
6ØØ LET I$=INKEY$:IF I$=""
    THEN GOTO 6ØØ
61Ø IF I$<"1" OR I$>"5" THEN
    GOTO 6ØØ
62Ø I=VAL(I$):ON I GOSUB
    1ØØØ,2ØØØ,3ØØØ,4ØØØ,
    5ØØØ
7ØØ IF A(M,2)<Ø THEN GOTO
    7ØØØ
71Ø ER=ER+INT
    (RND(1)*1ØØØ)-2ØØ
72Ø IF INT(RND(1)*16ØØ)
    -A(M,3)<Ø THEN GOSUB
    9ØØ

74Ø A(M,1) =A(M,2) +A(M,
    3)*ER
75Ø COLOR 15,1,1:
    CLS
79Ø NEXT M
8ØØ NEXT N
81Ø COLOR 15,1,1:
    CLS
82Ø PRINT "FIN DE
    JUEGO"
83Ø PRINT TAB(5);"ACTIVOS
    TOT.DE ";A$(1):PRINT
    TAB(11);"$";
    A(1,1)
84Ø IF NOP=2 THEN PRINT
    TAB(5);"ACTIVOS TOT. DE
    ";A$(2):PRINT
    TAB911);"$";A92,
    1)
85Ø PRINT TAB(2);"CUALQUIER
    TECLA PARA JUGAR DE
    NUEVO"
86Ø '
87Ø IF INKEY$="" THEN GOTO
    87Ø
88Ø RUN
9ØØ COLOR 15,1,1
9Ø5 JK=INT (RND(1)*
    1ØØ) + 5Ø:IF JK>
    A(M,3) THEN JK=
    A(M,3)
91Ø LOCATE 8,9:PRINT " R O B
    O "
92Ø PRINT " HAN SIDO
    ROBADOS ";JK;" KG DE
    ORO": A(M,3) =A(M,3)
    -JK:A(M,1) -A(M,1)
    -(JK*ER)
93Ø FOR X=1 TO 35:BEEP:NEXT
    X
94Ø COLOR 15,1,1:
    RETURN
```

ANALISIS DETALLADO DEL PROGRAMA

Para empezar, la línea 10 te pide que especifiques el número de jugadores y la línea 20 comprueba que la respuesta suministrada a la sentencia INPUT está dentro del margen permitido.

La línea 30 define p y nop con arreglo al número de jugadores.

En la línea 40 se dimensionan una serie de matrices juntamente con el precio del oro, er. La matriz a se utiliza para almacenar información acerca de los activos pertenecientes a cada uno de los jugadores y las minas; la matriz c se utiliza para almacenar información sobre las minas; la matriz a$ contiene los nombres de los jugadores y la r se utiliza para indicar si se han iniciado o no las labores de minería en la mina considerada por el jugador. La línea 50 inicializa los activos y el estado de la mina para ambos jugadores. Se asigna el valor 0 a r(1) y r(2) para indicar que todavía no se ha iniciado la minería en la primera mina en la que se van a hacer prospecciones. Otros valores asignados son los siguientes: a(1,1) y a(2,1) son los activos totales de cada jugador; a(1,2) y a(2,2) son los valores de efectivo de cada jugador; a(1,3) y a(2,3) son las cantidades de oro de cada jugador; a(1,4) y a(2,4) son los costes de la minería; a(1,5) y a(2,5) son los números de minas y finalmente a(1,6) y a(2,6) son las profundidades de cada mina. La línea 70 permite introducir el nombre de cada jugador.

El programa contiene un par de bucles FOR... NEXT, que empiezan en la línea 200 y terminan en las líneas 790 y 800. Estos son los bucles que definen el menú principal de opciones y la presentación de los ingresos de la compañía debidos a la minería, los costes de extracción, etc.

LAS VARIABLES N Y NOP

La variable n cuenta el número de pasadas realizadas por el jugador. La variable nop sirve para asegurarse de que ambos jugadores llegan hasta 30 pasadas. Más adelante el programa utiliza estas mismas variables para asegurarse de que se presentan los activos de ambos jugadores, etc.

En la línea 202 se definen los colores de la pantalla. La línea 210 presenta el título del juego: MINA DE ORO.

La línea 220 es la encargada de presentar el nombre o nombres de los jugadores. Cuando se elige la opción de dos jugadores, solamente se presenta el apellido.

Las líneas 230 a 300 presentan los valores de ACTIVOS TOTALES, CAJA EFECTIVO, CANTIDAD EN ORO, COSTES DE MINERIA, NUMERO DE MINAS, PROFUNDIDAD DE LA MINA Y PRECIO DEL ORO. Si hay dos personas jugando, se presentan los valores de ambos en los lugares apropiados, examinando los valores correspondientes a la variable nop.

La línea 400 presenta el nombre del jugador a quien le toca el turno en cada momento. Las líneas 500 a 540 le brindan al jugador las opciones Investigación y Desarrollo, Exploración e Informe, incremento de la profundidad de la mina en 200 metros, Precio del oro en dólares o pasar. La línea 550 invita al jugador a que teclee una instrucción.

Las líneas 600 a 620 utilizan la función INKEY$ para tener en cuenta la elección del jugador, comprobar que se trata de una elección válida y llamar a la subrutina encargada de su procesamiento.

La línea 700 examina si el valor total de los activos ha caído por debajo de cero, forzando un salto a la rutina de «final de juego» cuando así ocurre. En el próximo capítulo veremos la línea 7000 y siguientes. La línea 710 introduce fluctuaciones aleatorias en el precio del oro, por lo que tienes que



tener cuidado para vender tu oro en un momento en que su precio te permita hacer una operación favorable.

La línea 720 establece una comparación entre un número aleatorio y la cantidad de oro almacenada en tus cámaras acorazadas, al objeto de decidir si va a haber un robo o no; observa que son mayores las probabilidades de que ocurra un robo cuando tienes una gran cantidad de oro que cuando tienes una cantidad pequeña. La rutina de robo se extiende desde la línea 900 hasta la 940. La línea 905 elige la cantidad de oro que ha sido robada y la 920 se ocupa de presentar dicha cantidad en la pantalla.

La línea 740 calcula el valor total de los activos, sumando al valor de efectivo en caja el valor resultante del oro al precio vigente en cada momento. La línea 350 inicializa los colores de la pantalla, borrándola antes de que la instrucción NEXT envíe nuevamente el programa a la línea 200, dejándolo listo para la siguiente pasada.

Las líneas 810 a 840 constituyen la rutina de «juego terminado», que se utiliza cuando el activo total de uno de los dos jugadores ha caído por debajo de cero. La rutina presenta el estado financiero de ambos jugadores después de presentar el mensaje de FIN DE JUEGO.

Finalmente las líneas 850 a 880 corresponden a una rutina de ¿quieres jugar otra vez?

PARA OBTENER UN JUEGO MAS ADICTIVO

En el próximo capítulo veremos una serie de subrutinas que hacen que el juego resulte realmente adictivo. Veremos una rutina que permitirá reducir tus costes de minería por medio de la investigación y desarrollo, leer un informe relativo a las prospecciones realizadas en una mina, realizar excavaciones en la misma, escalonadas por etapas y cambiar tu oro por dólares.

Además veremos todos los datos que necesitarás para dibujar los gráficos que ilustran el estado de las minas de oro y el progreso realizado por las excavaciones.

IGUAL QUE EL REY MIDAS

- INVESTIGACION Y DESARROLLO DE NUEVOS METODOS EN MINERIA
- EXPLORACION DE NUEVAS MINAS
- EL INFORME SOBRE LA MINA
- PERFORACION DE LA MINA

Ha llegado el momento de que te hagas rico rápidamente. ¿Pero, has invertido en nuevas tecnologías antes de comenzar la exploración? ¿Cómo interpretas el resultado? ¿Y cuál es el mejor momento para vender? Tendrás que ser bastante perspicaz en la Mina de Oro.

Ya has visto en la primera parte de este juego cómo definir las distintas opciones que se ofrecen al jugador: Investigación y Desarrollo, Exploración e Informe, Aumento de la Profundidad de la Mina y Cambio de Oro por Dólares. Ahora puedes completar tu programa de la Mina de Oro con las subrutinas que manejan cada una de estas opciones.

Investigación y Desarrollo es la actividad que corresponde a la selección de la opción 1, Exploración Previa e Informe es la opción 2, Aumento de la Profundidad de la Mina es la opción 3 e Intercambio de Oro por Dólares es la opción 4. La opción 5 corresponde a pasar sin hacer nada, por lo que para ella no se requiere una subrutina completa. Las opciones 1, 2 y 4 introducen los elementos de aleatoriedad requeridos para que el juego se parezca de verdad al mundo real.

INVESTIGACION Y DESARROLLO

```
1000 COLOR 15,1,1:CLS
1010 LOCATE 4,3:PRINT
     "INVESTIGACION Y
     DESARROLLO":LOCATE 4,
     4:PRINT "(PARA
     DISMINUIR EL COSTE)"
1020 LOCATE 6,7:PRINT
     "CUANTO QUIERE";
     TAB(5); "INVERTIR
     ($)":INPUT RD
1050 A(M,4) =A(M,4)
     -INT(RD*.05)-1
1060 IF A(M,4)<0 THEN A(M,
     4)=0
1080 A(M,2)=A(M,2) -RD:A(M,
     1) =A(M,1)-RD
1100 LOCATE 3,13:PRINT "EL
     COSTE DE LA MINA";
     TAB(3);"SE HA REDUCIDO
     A $";INT(RD*.05) A+1;"
     POR 200M"
1110 FOR Z=1 TO 300::NEXT Z
1120 RETURN
```

La línea 1000 define los colores de la pantalla y borra ésta. A continuación la línea 1010 define la cabecera de pantalla antes de que la línea 1020 pregunte al jugador cuánto dinero quiere invertir en Investigación y Desarrollo; rd es la cantidad elegida.

La línea 1050 disminuye los costes de minería en una cantidad relacionada con el volumen de la inversión en Investigación y Desarrollo. En la línea 1060 se comprueba que los costes de minería no se hacen negativos. La línea 1080 ajusta los activos en metálico y totales para tener en cuenta la cantidad invertida en I + D.

La cantidad en que se ven reducidos los costes de minería se presenta en la línea 1100. La línea 1110 contiene un bucle FOR ... NEXT para introducir un pequeño retardo antes de que finalice la subrutina.

EXPLORACION E INFORME

```
2000 COLOR 15,1,1:CLS
2030 R(M)=0:C(M,1)=INT(RN
     D(1)*90)+10:C(M,2)=IN
     T((RND(1)*200:C(M,3)=I
     NT(RND(1)*200+1:LL=I
     NT(RND(1)*3)-1
2050 C(M,4) =C(M,2) +LL*200
2070 C(,5)=0: KK=INT
     (RND(1)*100):IF KK<C(M,
     1) THEN C(M,5)=1
2080 LOCATE 6,2:PRINT
     "INFORME CIENTIFICO
     ":LOCATE 2,5:PRINT
     "POSIB. ENCONTRAR
     ORO= ";C(M,1);
     "%":LOCATE 2,7:PRINT
     "PROFUNDIDAD
     ESTIMADA= ";C(M,2);
     "M":LOCATE 2,9: PRINT
     "CANTIDAD ESTIMADA=
     ";C(M,3);"KG."
2100 Z=INT (RND(1)*150000!):
     A(M,2)=A(M,2) -Z:A(M,1)
     =A(M,1)-Z
2110 LOCATE 0,12:PRINT "SE
     INICIAN EXCAVACIONES?
     (S/N)"
2120 R$=INKEY$:IF R$=""
     THEN GOTO 2120
2130 IF R$="S" THEN LET A(M,
     6)=0:R(M) = 1:GOTO
     3000
2500 RETURN
```

En la línea 2000 se borra la pantalla y se cambian sus colores. La línea 2030 pone a cero r(m) para indicar que todavía no ha empezado la excavación. En esta línea también se determina la probabilidad de encontrar oro, el valor esperado de la profundidad y la cantidad esperada. La variable ll es un número aleatorio comprendido entre 1 y −1, el cual se utiliza en la línea siguiente a fin de determinar la profundidad real del oro; recuerda pues que el valor de C(M,2) es precisamente el valor esperado de la profundidad.

La línea 2050 asigna a C(M,4) un valor igual a C(M,2) más o menos 200 metros (200 veces ll). Seguidamente la línea 2070 decide si la mina contiene realmente algo de oro. Se le asigna a C(M,5) el valor 0 para indicar que no hay oro. KK es un número aleatorio comprendido entre 0 y 99. KK se compara con la probabilidad de encontrar

oro, si KK es menor, entonces C(M, 5) se pone a uno para indicar que hay oro en la mina.

La línea 2080 presenta al jugador el informe científico de la mina. Aunque al jugador se le dice la probabilidad de que encuentre oro y la profundidad más probable, el que esto ocurra realmente depende de varios factores aleatorios. En consecuencia deberás utilizar tu propio juicio para determinar si la inversión merece la pena o no.

Ocupémonos ahora de las malas noticias: el informe hay que pagarlo. Es imposible predecir lo que costará, pero puede ser un valor comprendido entre nada y 150.000 dólares; éste es el valor de Z elegido en la línea 2100. El coste de la exploración y el informe hay que deducirlo de la disponibilidad en efectivo, y esta deducción aparecerá también en los activos totales.

En este momento al jugador se le ofrece la posibilidad de iniciar las excavaciones. En la línea 2110 se hace la pregunta ¿SE INICIAN LAS EXCAVACIONES? Si la respuesta es sí, el programa salta a la rutina de minería que empieza en la línea 3000.

EXCAVACION

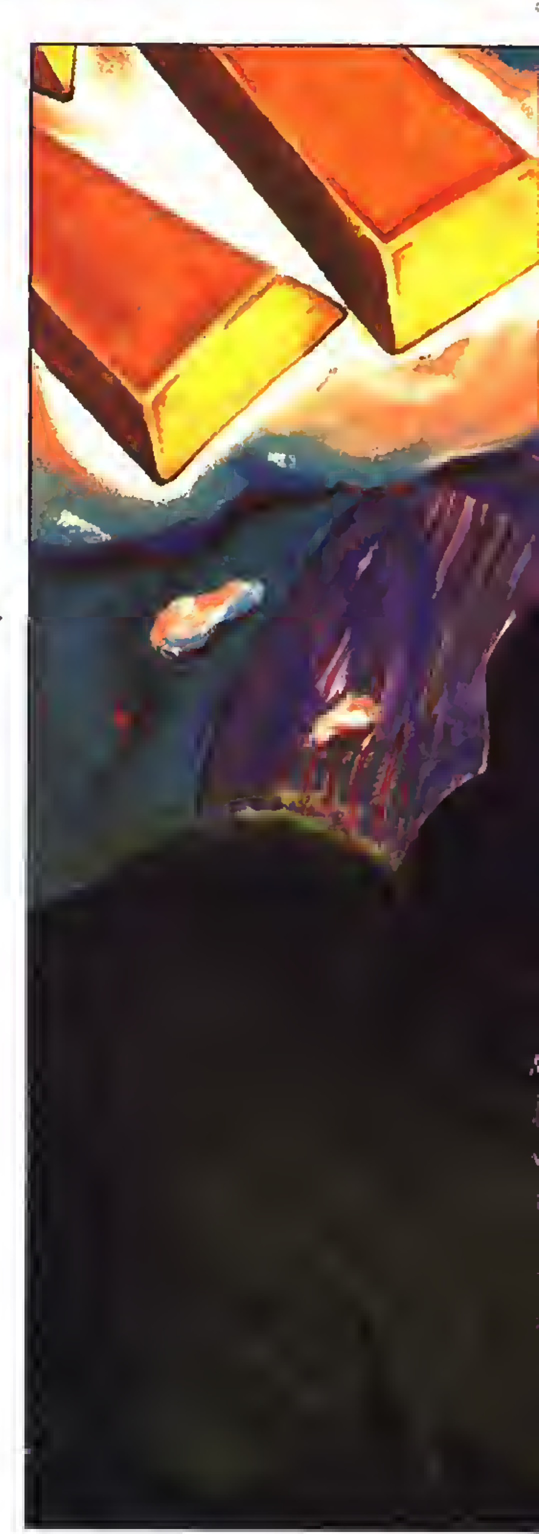
```
3ØØØ COLOR 15,1,1:CLS
3Ø1Ø IF R(M)=Ø THEN LOCATE
     2,9:PRINT "NO HAS
     EXPLORAOO
     TODAVIA":FOR Z=1 TO
     1Ø:BEEP:NEXT Z:RETURN
3Ø2Ø COLOR 15,1,1:CLS
3Ø22 PRINT TAB(14);CHR$(14
     7);CHR$(148);CHR$(149
     );TAB(14);CHR$(15Ø);CH
     R$(151);CHR$(152);CHR
     $(153);TAB(13);CHR$(1
     54);CHR$(155);CHR$(15
     6);CHR$(157);CHR$(158
     );TAB(31);CHR$(32)
3Ø25 FOR Z=1 TO 32:PRINT
     CHR$(144);:NEXT Z
3Ø6Ø LOCATE Ø,4:FOR Z=1ØØ
     TO 14ØØ STEP
     1ØØ:LOCATE Ø,3+Z/1ØØØ:
     PRINT Z :NEXT Z
3Ø9Ø A(M,2)=A(M,2) -A(M,
     4):A(M,1) =A(M,1)-A(M,
     4): A(M,6)=A(M,6)
     +2ØØ:FOR XX=1 TO
     2ØØ:NEXT XX
31ØØ LOCATE 15,3: PRINT
     CHR$(146):FOR F=4 TO
     (A(M,6)/1ØØ)+3:LOCATE
     15,F:PRINT
     CHR$(145):FOR W=1 TO
     1Ø:BEEP:NEXT W:NEXT F
312Ø IF A(M,6)=C(M,4) AND
     C(M,5)=1 THEN GOTO
     35ØØ
313Ø LOCATE 2,6:PRINT "NO
     ENCONTRASTE ORO
     TODAVIA":IF A(M,6)=C(M,
     2)+2ØØ THEN LOCATE Ø,
     18:PRINT "ESTA MINA NO
     TIENE ORO.PRUEBA CON
     OTRA.":FOR Z=1 TO
     1Ø:BEEP:NEXT Z:A(M,
     6)=Ø:R(M)=Ø
314Ø FOR XX=1 TO 3ØØ:NEXT
     XX
33ØØ RETURN
35ØØ LOCATE 12,F:PRINT
     "ORO":FOR Z=-2Ø TO
     5Ø:BEEP:NEXT Z:FOR
     XX=1 TO 5ØØ:NEXT XX
355Ø A(M,5)=A(M,5) +1:A(M,
     3)=A(M,3) +C(M,3):A(M,
     1) =A(M,1)+(A(M,3)*ER) :
     A(M,Y)=Ø:R(M)= Ø:GOTO
     33ØØ
```

Puedes acceder a esta rutina desde dos sitios del programa. Como ya has podido ver, se te ofrece la opción de iniciar las excavaciones desde la rutina de Exploración e Informe. Pero también se utiliza cuando optas por aumentar la profundidad de la mina en 200 metros, eligiendo el número 3 de la lista de opciones.

Como de costumbre, la primera línea de la rutina se limita a borrar la pantalla y a definir sus colores. La línea 3010 comprueba que se ha completado la fase de exploración para que pueda iniciarse la fase de excavaciones. La línea 3020 vuelve a ocuparse de nuevo de la pantalla, dejándola lista para nuevas presentaciones.

Las líneas 3022 a 3090 se ocupan de los gráficos que muestran en pantalla la mina de oro. La línea 3100 ilustra la excavación y genera algunos efectos sonoros.

La línea 3120 examina si la excavación ha llegado al nivel donde se encuentra el oro y si hay oro en la mina



(podría ser que se llegara al nivel esperado para el oro y ocurriese que la mina no contiene absolutamente nada). Si se llega hasta el oro, el programa salta a la línea 3500, que informa al jugador de que se ha encontrado. La línea 3550 ajusta ahora el valor de los activos del jugador, con arreglo al valor del oro encontrado.

Si no hay oro, el programa continúa hasta la línea 3130. Si la excavación ha sobrepasado la altura esperada para el oro en 200 metros, el jugador es informado de que la mina no lo contiene. Si la excavación aún no ha llegado tan lejos, el jugador recibe el mensaje siguiente: NO HAY ORO TODAVIA.

EL PRECIO DEL ORO

```
4ØØØ COLOR 15,1,1:
     CLS
```



```
4020 LOCATE 7,2:PRINT
     "AGENCIA DE CAMBIO
     ":LOCATE 0,6:PRINT "EL
     TIPO DE CAMBIO ACTUAL
     ES:-":LOCATE 5,8:PRINT
     "1 KG DE ORO= $";
     ER:LOCATE 2,12:PRINT
     "CUANTOS KG QUIERES
     CAMBIAR":INPUT
     NTE
4070 IF NTE>A(M,3) THEN
     LOCATE 0,16:PRINT "NO
     HAY TANTO
     ORO"
4080 NTE=INT(NTE)
4090 IF NTE>A(M,3) OR NTE<0
     THEN GOTO
     4020
4095 LOCATE 0,16:PRINT
     CHR$(32); TAB(31);
     CHR$(32)
4100 A(M,3)=A(M,3)
     -NTE:A(M,2)=A(M,2)
     +(NTE*ER):A(M,1) =A(M,
     1)+(NTE*ER)
4130 LOCATE 1,16:PRINT "KG
     CAMBIADOS POR $";
     NTE*ER:FOR XX=1 TO
     300:RETURN
5000 RETURN
```

La línea 4000 configura el programa.

La línea 4020 presenta el título en la pantalla, el valor de mercado del oro y los mensajes para seleccionar el número de kilogramos que hay que vender. La línea 4070 comprueba si posees suficiente oro. La línea 4080 sirve para asegurarse de que la cantidad de oro vendido es un número entero.

La línea 4090 vuelve a enviar el programa al punto de presentación de mensajes si la cantidad que se pretende vender supera a la cantidad de oro que se posee,o es menor que cero. La línea 4100 modifica el valor de los activos totales con arreglo a la cantidad de oro vendido.

Esta subrutina informa al jugador de cuánto oro se ha vendido y cuántos dólares se han recibido a cambio, cosa que se hace en la línea 4130. La línea 5000 corresponde a la opción de pasar sin hacer nada.





LOS TOQUES FINALES

```
1 FORN=32ØØ 3312+7:READ A♀OKE N,A:NEXT N
7ØØØ COLOR 15,1,1: CLS
7Ø1Ø LOCATE 12,9:PRINT A$(M):LOCATE 8,1Ø:PRINT "HA HECHO BANCARROTA":LOCATE 1,2Ø:PRINT "PULSA UNA TECLA PARA JUGAR OTRA VEZ"
7Ø3Ø DEFUSR=&H9F: X=USR(Ø): RUN 5
8ØØØ DATA 255,85,17Ø,Ø,Ø,Ø,Ø,Ø,62,28,56,126,28,62,12Ø,28
8Ø1Ø DATA 255,255,62,126,127,6Ø,124,126,Ø,Ø,Ø,Ø,1,1,1,1
8Ø2Ø DATA 7,29,49,45,255,255,91,126,128,96,48,8Ø,152,14Ø,252,138
8Ø3Ø DATA 1,1,1,49,49,49,49,255,122,187,62,95,153,255,153,126
8Ø4Ø DATA 2Ø9,177,224,128,128,128,128,128,Ø,Ø,128,128,64,32,32,16
8Ø5Ø DATA 1,3,7,7,4,4,7,7,255,255,255,255,149,149,159,159
8Ø6Ø DATA 24,126,153,255,126,153,126,219,128,192,224,24Ø,249,168,248,255
8Ø7Ø DATA 16,8,8,4,14,31,31,255
```

Las líneas 7000 a 7030 contienen una rutina para jugar otra vez el juego.

Las líneas 8000 a 8070 contienen los DATA para los GRAFICOS.

¡Ahora ya puedes amasar tu inmensa fortuna y comprar todas esas cosas que siempre te has prometido adquirir un día!

DOMINANDO EL TABLERO

Se levanta el telón para la presentación del Juego OTELO. Programa este sencillo juego de estrategia y de engaño y desafía a tu ordenador. Pero, ¡cuidado!, no es tan sencillo como parece.

OTELO es un juego de estrategia que se juega sobre un tablero de ocho por ocho casillas —un tablero de ajedrez o de damas puede servirnos—. Las reglas son muy simples y el juego cuenta con varios trucos.

El objetivo consiste en capturar o *comer* el mayor número de fichas posibles a tu oponente. Cada jugador ha de colocar una ficha en el tablero hasta llenarlo por completo. Cada jugador comienza con dos fichas y ha de intentar capturar las del jugador contrario *rodeándoselas*. Ello se logra colocando una ficha extra al final de una fila de fichas, de manera que el adversario se vea acorralado por tus fichas. Todas las fichas adversarias que hayas acorralado serán reemplazadas ahora por fichas tuyas.

El número de puntos será simplemente el número de fichas de cada jugador que haya sobre el tablero durante cada jugada. El ganador será aquel que consiga tener el mayor número de piezas sobre el tablero cuando éste esté lleno.

En esta versión, tú juegas contra el ordenador, el cual también muestra en pantalla el tablero y lleva la puntuación.

CONSEJOS Y TRUCOS

Al igual que en cualquier otro juego de estrategia, éste también cuenta con algunos trucos que pueden serte de gran ayuda. Si ésta es la primera vez que juegas al OTELO, los siguientes consejos te serán muy útiles.

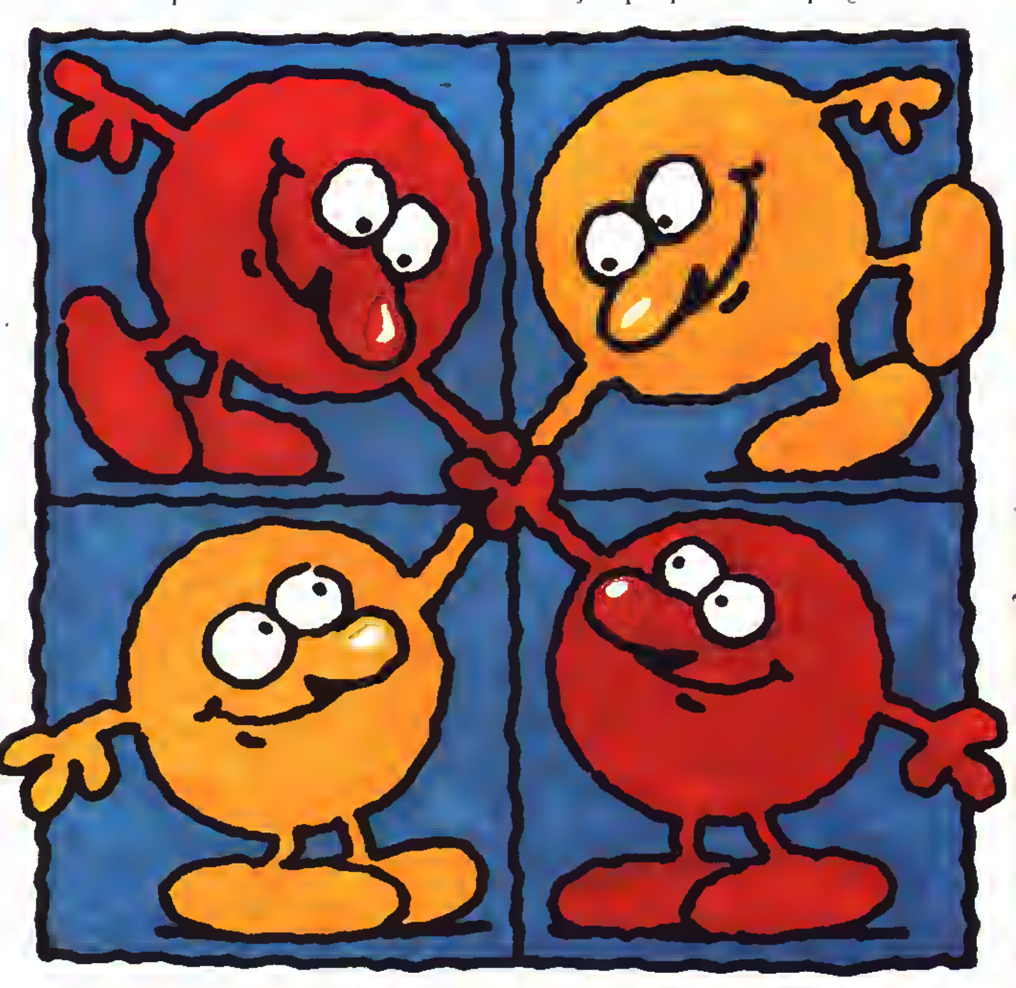
Las fichas de los extremos son muy valiosas, pues no pueden ser recuperadas una vez han sido capturadas —la razón de ello es debido a que estas fichas no pueden ser acorraladas como las de otras posiciones del tablero—. Como consecuencia, pueden ser muy significativas para ganar, y es muy importante capturar las esquinas, incluso sacrificando un movimiento que podría habernos proporcionado una mayor puntuación. Asimismo, también son intocables cada una de las fichas emplazadas junto a las fichas de las esquinas.

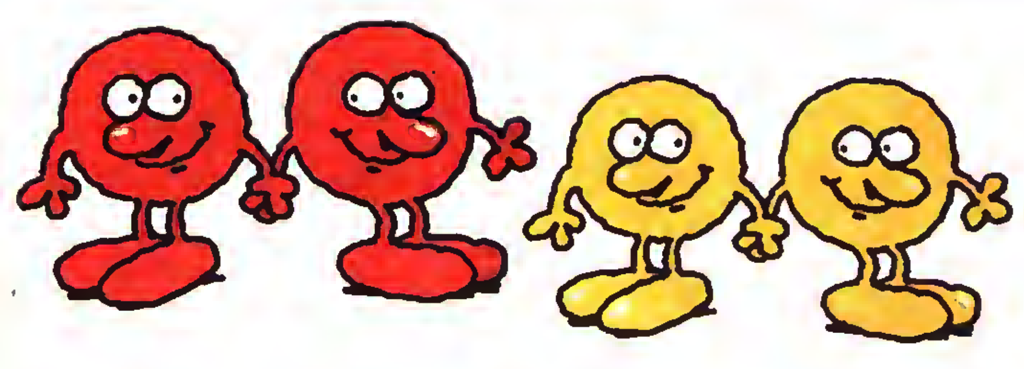
Puesto que una pieza puede enlazar más de una línea —arriba, abajo y en diagonal—, el movimiento más obvio no siempre será el mejor, mientras que en las últimas fases del juego, a menudo tú puedes enlazar dos o tres líneas añadiendo una sola ficha.

Debes pensar siempre con anticipación al adversario. Tal vez puedas conseguir engañar a tu oponente creando situaciones favorables para ti —para conquistar posiciones vitales— simulando una mala movida de tus fichas.

EL PROGRAMA

El programa juega el papel de tu oponente —con fichas negras—, y ya verás cómo un programa, comparativamente sencillo, es capaz de jugar con mucha destreza el desafiante juego OTELO. Una de las grandes ventajas que presenta el programa de





ordenador OTELO, en comparación a un juego normal de tablero, es que en la versión de ordenador te evitas toda la parte del trabajo difícil. El ordenador te evita el tener que volver las fichas o sustituirlas por otras de diferente color. Tu único trabajo consiste en concentrarte en el juego.

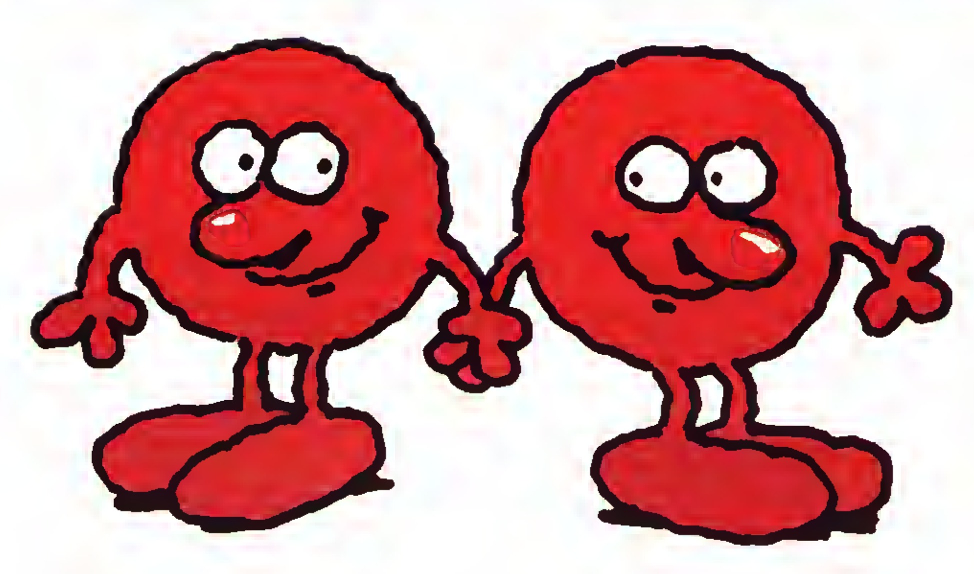
Puesto que el ordenador considera todas las posibilidades, tarda un poco en realizar sus movimientos de fichas, aunque se vuelve más veloz a medida que avanza el juego y quedan cada vez menos casillas vacías.

```
170 COLOR 1Ø:SCREEN Ø,Ø,Ø
180 VPOKE 2752,&BØØØØØØØØ
190 VPOKE 2753,&BØØØØØØØØ
200 VPOKE 2754,&BØØ11ØØØØ
210 VPOKE 2755,&BØ1111ØØØ
220 VPOKE 2756,&BØ1111ØØØ
230 VPOKE 2757,&BØØ11ØØØØ
240 VPOKE 2758,&BØØØØØØØØ
250 VPOKE 2759,&BØØØØØØØØ
260 VPOKE 268Ø,&BØØØØØØØØ
270 VPOKE 2681,&BØØØØØØØØ
280 VPOKE 2682,&BØØ11ØØØØ
290 VPOKE 2683,&BØ1ØØ1ØØØ
300 VPOKE 2684,&BØ1ØØ1ØØØ
310 VPOKE 2685,&BØØ11ØØØØ
320 VPOKE 2686,&BØØØØØØØØ
330 VPOKE 2687,&BØØØØØØØØ
340 VPOKE 2416,&B111111ØØ
350 VPOKE 2417,&B1ØØØØ1ØØ
360 VPOKE 2418,&B1ØØØØ1ØØ
370 VPOKE 2419,&B1ØØØØ1ØØ
380 VPOKE 242Ø,&B1ØØØØ1ØØ
390 VPOKE 2421,&B1ØØØØ1ØØ
400 VPOKE 2422,&B1ØØØØ1ØØ
410 VPOKE 2423,&B111111ØØ
420 GOTO 114Ø
43Ø PRINT "Yo muevo"
44Ø S=O:T=X:H=Ø
45Ø FOR A=2 TO 9:FOR B=2 TO
    9
46Ø IF A(A,B)<>46 THEN 61Ø
47Ø Q=Ø
48Ø FOR C=-1 TO 1:FOR
    D=-1 TO 1
49Ø K=Ø:F=A:G=B
5ØØ IF A(F+C,G+D)<>S THEN
    53Ø
51Ø K=K+1:F=F+C: G=G+D
52Ø GOTO 5ØØ
53Ø IF A(F+C,G+D)<>T THEN
    55Ø
54Ø Q=Q+K
55Ø NEXT D:NEXT C
56Ø IF A=2 OR A=9 OR B=2 OR
    B=9 THEN Q=Q*2
57Ø IF A=3 OR A=8 OR B=3Ø
    OR B=8 THEN Q=Q/2
58Ø IF (A=2 OR A=9) AND
    (B=3 OR B=8) OR (A=3 OR
    A=8) AND (B=2 OR B=9)
    THEN Q=Q/2
59Ø IF Q<H OR
    ((RND(1)-1)+1<.3 AND
    Q=H) THEN 61Ø
6ØØ H=Q:M=A:N=B
61Ø NEXT B:NEXT A
62Ø IF H=Ø AND R=Ø THEN
    1Ø9Ø
63Ø IF H=Ø THEN 65Ø
64Ø GOSUB 98Ø
65Ø GOSUB 77Ø
66Ø INPUT "Cual es tu
    movimiento";R
67Ø '
68Ø S=X:T=O
69Ø IF R=Ø THEN 75Ø
7ØØ IF R<11 OR R>88 THEN
    66Ø
71Ø R=R+11
72Ø M=INT(R/1Ø)
73Ø N=R-1Ø*M
74Ø GOSUB 98Ø
75Ø GOSUB 77Ø
76Ø GOTO 43Ø
77Ø '
```



```
780Ø C=Ø:H=Ø
790Ø CLS
800Ø PRINT
810Ø PRINT "     La computadora
        es X"
820Ø PRINT "         El humano
        es O"
830Ø PRINT
840Ø PRINT TAB(3);"12345678"
850Ø FOR B=2 TO 9:PRINT B-1;
860Ø FOR D=2 TO 9
870Ø PRINT CHR$(A(B,D));
880Ø IF A(B,D)=X THEN C=C+1
890Ø IF A(B,D)=O THEN H=H+1
900Ø NEXT D
910Ø PRINT B-1
920Ø NEXT B
930Ø PRINT TAB(3);"12345678"
940Ø PRINT :PRINT
950Ø PRINT "Computadora:"C
960Ø PRINT:PRINT "Humano:"H
970Ø RETURN
980Ø FOR C=-1 TO 1:FOR
        D=-1 TO 1
990Ø F=M:G=N
1ØØØ IF A(F+C,G+D)<>S THEN
        1Ø3Ø
1Ø1Ø F=F+C:G=G+D
1Ø2Ø GOTO 1ØØØ
1Ø3Ø IF A(F+C,G+D)<>T
        THEN 1Ø7Ø
1Ø4Ø A(F,G)=T
1Ø5Ø IF M=F AND N=G
        THEN 1Ø7Ø
1Ø6Ø F=F-C:G=G-D: GOTO
        1Ø4Ø
1Ø7Ø NEXT D:NEXT C
1Ø8Ø RETURN
1Ø9Ø GOSUB 77Ø
11ØØ IF C>H THEN PRINT "Soy
        soy el mejor, yo gano"
111Ø IF H>C THEN PRINT "Me
        has vencido."
112Ø IF H=C THEN PRINT
        "Hemos empatado"
113Ø END
114Ø ' CLS
115Ø X=ASC("X"): O=ASC("O")
116Ø DIM A(1Ø,1Ø)
117Ø FOR B=1 TO 1Ø:FOR C=1
        TO 1Ø
118Ø IF B<>1 AND C<>1 AND
        B<>1Ø AND C<>1Ø
        THEN A(B,C)=ASC(".")
119Ø NEXT C:NEXT B
12ØØ A(5,5)=X:A(6,6)=X
121Ø A(6,5)=O:A(5,6)=O
122Ø P=Ø
123Ø PRINT "Quieres empezar
        tu?"
124Ø PRINT "S / N"
125Ø INPUT W$
126Ø GOSUB 77Ø
127Ø IF W$="S" OR W$="s"
        THEN GOTO 66Ø
128Ø GOTO 43Ø
```





La línea 170 prepara el color de la pantalla y acciona el screen 0. Desde las líneas 180 a 410 se diseñan las formas de las fichas utilizando la memoria de vídeo.

Por su parte, la línea 420 lleva el control del ordenador a la línea 1140; desde aquí hasta la línea 1220 se prepara la variable de matriz «A» con el fin de obtener el aspecto inicial del tablero.

Desde la línea 1230 hasta la 1280 se ofrece la posibilidad de que el jugador comience el juego o de que lo haga la máquina; en caso de que hayamos elegido comenzar, el ordenador traslada su control a la línea 660; en caso contrario, el control pasará a la línea 430.

A continuación, la línea 660 nos preguntará cuál es nuestro movimiento y guardará éste en la variable «R»; la línea 750 se asegura de que no optemos por introducir «0» dando a entender al ordenador que pasamos.

En seguida, las líneas comprendidas entre 790 y 970 se encargan de mostrar después de cada jugada el aspecto actual del tablero.

Entre las líneas 800 y 830 se muestran la ficha del jugador y la del ordenador.

El programa continúa con las líneas comprendidas entre 840 y 900, que dibujan el tablero y contabilizan el número de puntos de cada jugador. Por su parte, las líneas 950 y 960 muestran el número de puntos de cada jugador.

En la línea 1100 comienza la rutina de fin de juego; en ella se elige al vencedor, consultando el número de fichas de cada jugador. Al final del juego se pueden obtener tres resultados: elegir como ganador al jugador, elegir al ordenador, o bien empatar.

JUGANDO AL OTELO

Cuando ejecutes el programa, el ordenador te preguntará si deseas mover tú primero. Cada vez que hagas un movimiento tendrás que introducir dos coordenadas. Éstas indicarán tu posición dentro de una escala entre uno y ocho —los números de líneas y de columnas se editan de arriba abajo a un lado del tablero—. Las coordenadas se introducen indicando primero la posición horizontal (línea) y luego la vertical (columna).

El programa no reconoce a un compañero de juego poco hábil, ni tampoco es capaz de adivinar si estás aburrido, así pues, introduciendo un 0 como coordenada, se da fin al juego. Othello llegará a convertirse en un pasatiempo absorbente: un juego que se aprende en un minuto pero que para llegar a dominarlo se tarda toda una vida.

LA LUNA A TUS PIES

En este formidable juego vas a necesitar de toda la habilidad y sangre fría de que seas capaz para maniobrar el módulo lunar de forma que pueda alunizar perfectamente.

La programación de juegos no tiene por qué generar complicados programas para producir un juego independiente y completo. Aquí te presentamos una versión del célebre programa *Módulo lunar (Lunar lander)* que contiene gráficos en alta resolución y un control total sobre el aparato.

El juego es completo y muy variado según lo presentamos, pero tú tienes la posibilidad de adaptarlo a tus preferencias personales. Por ejemplo, puede que te guste añadir una rutina del tipo «¿otro intento?» para evitar volver otra vez al RUN una vez que se ha concluido el descenso. O bien puede que desees alterar los gráficos y los sonidos. Todo depende de ti.

CONTROLES

Deberás utilizar las teclas del cursor 5, 7 y 8.

```
1 CLS: RESTORE 5004: FOR N=1
  TO 33
2 READ Q:
  Q$=Q$+CHR$(Q):NEXT N
3 PRINT: FOR N=1 TO 10: PRINT
  TAB(5);Q$:PRINT:NEXT N
4 FOR N=1 TO 400: NEXT N
7 COLOR 15,1,4: CLS
8 SCREEN 2: OPEN "GRP:"AS1
10 RESTORE 4500: FOR N=1 TO
   8: READ W:
   S$=S$+CHR$(W):NEXT N
11 SPRITE$(0)=S$
20 FOR N=1 TO 50
21 ZZ=RND(1):ZZ=ZZ*255
22 ZY=RND(1): ZY=(ZY*135)+40
25 PSET(ZZ,ZY):NEXT N
70 PSET(0,175): RESTORE 80:
   FOR N=1 TO 16
71 READ GX,GY:LINE
   -(GX,175-GY)
72 NEXT N
80 DATA 18,30,18,-15,18,-8,18,8
81 DATA 16,20,16,5,13,-20,16,-8
82 DATA 18,-4,15,0,10,10,20,2
83 DATA 5,10,-20,10,-10,20,-5,
   18,20
90 PRESET(32,0): PRINT#1,
   "FUEL:"
91 PRESET(144,0): PRINT#1,
   "VELOCIDAD:"
110 LX=RND(1): LX=(LX*240)
    +10
111 LY=RND(1): LY=160-(15+
    (LY*10))
112 XV=RND(1): XV=XV*15
113 YV=0: F=246
115 GOSUB 4000
120 GOSUB 1000: GOSUB 2000:
    GOSUB 3000
130 IF LY>20 THEN GOTO 120
135 FOR N=1 TO 100: NEXT N
140 IF LX<154 OR LX>164 OR
    ABS(YV)>4 THEN GOTO 160
150 PRINT#1,"FELICIDADES LO
    HAS LOGRADO!!": RESTORE
    5000
151 FOR N=1 TO 14
152 READ A,B: BEEP
153 NEXT N: GOTO 170
160 PRINT#1,"!!!CRASHED¡¡¡"
161 FOR T= TO 50
162 RD=RND(1): RD=(RD*15)+1
163 COLOR,,RD: BEEP: NEXT T
```

```
170 FOR N=1 TO 800: NEXT N
180 END
1000 IF LY<160 THEN
     GOSUB 4000
1010 LX=LX+XV: LY=LY+YV
1011 IF LY<300 THEN BEEP
1030 IF LX<5 THEN LX=LX-242
1035 IF LX>250 THEN LX=
     LX-242
1036 IF LY>160 THEN RETURN
1037 IF LY<10 THEN RETURN
1040 GOSUB 4000
1050 RETURN
2000 YV=YV-.5: IF F<1 THEN
     RETURN
2010 D$=INKEY$
2011 IF D$="7" AND F>3 THEN
     YV=YV+1: F=F-3: RETURN
2020 IF D$="5" THEN XV=XV-1:
     F=F-1: RETURN
2030 IF D$="8" THEN XV=XV+1:
     F=F+1: RETURN
2040 RETURN
3000 PRESET(80,0): COLOR 15:
     PRINT#1," "+STR$(F)+" "
3001 GOSUB 6000: YV1=INT(YV)
3002 PRESET(224,0): COLOR 15:
     PRINT#1," "+STRS(YV1)+"
     "
3003 GOSUB 6010: COLOR 15
3010 RETURN
4000 PUT SPRITE 0,(LX,LY),8
4010 RETURN
4500 DATA 24,36,36,60,126,126,
     66,129
5000 DATA .2,4,.2,7,.2,5,.2,12
5001 DATA .2,0,.2,4,.2,4,.2,5
5002 DATA .6,7,.2,12,.2,0,.2,4
5003 DATA .2,2,.6,0
5004 DATA 42,42,32,98,121,32,32,
     68
5005 DATA 65,78,73,69,76,32,67,65
5006 DATA 76,86,69,84,32,76,69,
     80
5007 DATA 69,75,72,73,78,69,32,
     42,42
6000 PRESET(80,0): COLOR 1:
     PRINT#1,"******"
6001 RETURN
6010 PRESET(224,0): COLOR 1:
     PRINT#1,"****"
6011 RETURN
```



JUGANDO A LA GUERRA: PRIMEROS PASOS

Diseñar un juego de guerra para ordenador es un proyecto fascinante. Descubre lo que ello implica y moviliza, entonces, tu microordenador para empezar a introducir el juego de batalla táctico-terrestre de INPUT.

Con la llegada de las computadoras, los juegos de guerra han alcanzado realmente la mayoría de edad. A pesar de que dichos juegos existen desde hace miles de años, jugar a ellos ha sido hasta hace poco un incómodo proceso que involucraba un espacio de terreno y completas falanges de pequeños modelos. Un ordenador resguarda todo al abrigo del polvo, presenta mapas en una pantalla-base y puede ser siempre tu oponente.

Los juegos de guerra tienen, por supuesto, un serio y mortal propósito, enseñar al militar cómo combatir en las guerras, más provechosamente. Y aunque muchos juegos de guerra tienen, como principal objetivo, el aniquilamiento del enemigo, es realmente cierto que se ha de tener un gran interés para crear y participar en juegos de guerra por ordenador. Este interés, sin embargo, es más afín para el que posee experiencia en el juego del ajedrez, que es el no va más de los juegos de enfrentamiento.

En pantalla se representa un mapa del campo de batalla mostrando generalmente las posiciones que ambas partes ocupan. En cualquier caso, un juego de ordenador puede tener un grado de realismo mayor que los juegos tradicionales, ya que es posible, si tú lo deseas, tener las unidades del oponente desplegadas sobre el tablero, pero ocultas hasta que sus posiciones han sido descubiertas. Los juegos de guerra por ordenador pueden ser para dos jugadores, o también, como el que verás desarrollado en INPUT, un juego en el cual una de las partes está controlada por el ordenador.

El juego, por supuesto, simulará cada aspecto de una guerra real, con la mayor fidelidad posible. En un juego de un home-computer, tú no podrás reproducirlo todo con exactitud. Ni tampoco lo desearás –el resultado de incluirlo todo podría ser un juego tan horriblemente realista que vosotros, jugadores, no hallaríais en él, el entretenimiento necesario para querer jugar.

DISEÑANDO UN JUEGO DE GUERRA

La guerra consiste en el movimiento de «unidades de combate» (que generalmente son seres humanos, pero que también pueden ser tanques o armas sobre el campo de batalla), hasta que alcanzan a las unidades enemigas, cuando intentan forzar al oponente a que se someta, casi siempre por combate. Los ingredientes básicos de un juego de guerra, son pues, dos ejércitos, sus movimientos y el combate entre las dos unidades enfrentadas.

Partiendo de esto como base, puedes considerar el extenso campo de diseño de tu juego de guerra. ¿Ocupará el juego zonas de tierra, mar o de aire? ¿Será un juego estratégico de larga escala con muchos ejércitos, un pequeño juego táctico con dos ejércitos sobre un único campo de batalla, o una refriega entre combatientes individuales?

El período histórico tendrá una gran influencia sobre el tipo de juego: ¿de qué clase de tecnología (si la hay) dispondrán los combatientes, y qué combatientes van a ser?

Tal vez, desees intentar revivir alguna batalla histórica, tal vez intentes forzar al ejército de Napoleón a atravesar Moscú, o quizá desees inventarte tu propio conflicto. Aquí, tú puedes dejar correr tu imaginación con toda libertad.

El siguiente paso consistirá en pensar en las «reglas» de un juego de guerra, tomando decisiones sobre todos los detalles que quieras incluir en tu juego y en qué circunstancias éstos ayudarán o perjudicarán a tu ejército. Por ejemplo, tal vez quieras incluir en tu juego las armaduras: el empleo de armaduras puede proteger al soldado durante el combate, pero ello retrasaría su movimiento al combatir (o en la retirada después de la lucha).

El juego podrá incluir tantos componentes de este tipo como quieras imaginar, para lograr un juego de guerra más realista. Pero la memoria de tu ordenador estará limitada por el número de diferentes puntos de detalle que incluyas y por lo complejas que hagas tus reglas.

Lo mejor es ceñirse a los componentes más importantes:

– Información geográfica sobre dónde están las tropas y qué tipo de terreno ocupan: ¿cómo afectará éste al movimiento de las tropas y cómo quedarán éstas cubiertas?
– Tropas: ¿qué cantidad hay? ¿qué armas y armaduras poseen? ¿qué rápido pueden moverse?
– Movimiento: considerar las reglas sobre cuán lejos pueden moverse las unidades y cómo afecta el terreno al movimiento.
– Ordenes: un mecanismo para dar órdenes y tal vez, alguna posibilidad de que las tropas puedan desobedecerlas.
– Elección de computador: ¿cuánta inteligencia poseerá el computador?
– Combate: ¿qué tipo? El combate generalmente está dividido en combate de misiles y combate cuerpo a cuerpo. Los misiles pueden ser desde los lanzados hacia unos ejes, a los misiles intercontinentales

Una vez que el tipo, período y componentes del juego han sido determinados, necesitarás considerar cómo van a estar representados en tu microordenador. Aquí hay dos aspectos



a tener en cuenta: cómo se manifestará el juego al jugador y cómo al ordenador. En esta serie de varios artículos, verás qué es lo que hay que tener en cuenta para construir un pequeño juego de guerra. Se trata de una batalla táctica terrestre en la que combaten dos ejércitos medievales. No obstante, el juego no se encuentra ubicado en ningún período en particular. Al igual que muchos juegos de guerra de ordenador, la pantalla muestra un mapa representando las posiciones de los dos ejércitos y el terreno.

Los jugadores (en este caso, tú y el ordenador) actúan como los jefes de sus propias unidades, y deben tomar decisiones de estrategia y dar órdenes apropiadas a sus hombres.

Todas las instrucciones serán dadas cuando el programa esté completado en la última parte de este artículo, pero generalmente, al jugador se le da la opción de emitir nuevas órdenes para cada unidad o dejarlas tal y como están. El juego sigue organizando por turnos, la disposición de las tropas, lo cual puede o no acabar en conflicto. La victoria en cualquier conflicto estará determinada por el tipo relativo y por la fuerza de los combatientes –además de una cierta cantidad de suerte. El juego continúa hasta que un jugador ha reducido las fuerzas del otro a un insostenible nivel.

Este juego presentará importantes gráficos de información –el mapa con los ejércitos– de forma continuada, pero reserva un área para mostrar el texto de instrucciones temporales y objetivos.

Es mejor usar gráficos definidos de usuario para visualizar el mapa. De este modo, es sencillo manejar las dos áreas de display utilizando el texto del manual de instrucciones para ambos.

En este juego hay cuatro clases de terreno –llanuras, pueblos, bosques y colinas.

Los espacios blancos pueden ser utilizados para representar llanuras, de manera que no sean necesarios los gráficos. Se necesitarán gráficos para cada área, a fin de dar una clara representación de colinas, pueblos, unidades y bosques.

Cada ejército tiene unidades: un jefe con sus caballeros, un segundo cuerpo de caballeros (sargentos), dos unidades de hombres de armas, dos unidades de arqueros, y dos de aldeanos.

Esto da un total de nueve gráficos diferentes: número uno, pueblos; número dos, bosques; números tres y cuatro, colinas; número cinco, jefe (representado por una bandera); número seis, caballeros (una maza); número siete, hombres de armas (un escudo); número ocho, arqueros (un arco); y número nueve, aldeanos (una espada).

Esta sección coloca los nueve gráficos.

```
230 FOR I=520 TO 591
240 READ B
250 VPOKE I,B
290 NEXT I
2570 DATA 16,16,60,126,255,189,
     231,231
2590 DATA 16,56,84,16,56,84,146,
     16
2610 DATA 8,20,34,65,6,8,16,224
2630 DATA 0,48,72,132,2,0,0,0
2650 DATA 128,240,255,252,143,
     128,128,128
2670 DATA 64,240,72,68,68,68,78,
     68 ,
2690 DATA 255,231,231,129,129,
     231,102,60
2710 DATA 249,70,38,25,9,5,3,1
2530 DATA 1,2,4,8,16,160,64,160
```

LO QUE MUESTRA LA PANTALLA

Estas pocas líneas son necesarias para limpiar el área asignada para la impresión de tus instrucciones:

```
2540 REM
2550 FOR K=17 TO 21
2551 LOCATE 0,K: PRINT "
2552 NEXT K
2555 RETURN
```

El programa trata el visor de la pantalla como dos ventanas –una ventana de texto y una ventana de mapa.

La ventana de texto tiene que ser limpiada y reescrita con frecuencia durante el juego, pero la ventana para el mapa tiene un display constante, con pequeños movimientos ocasionales de las unidades que muestra.

Cada ordenador ve la pantalla como una única sección continua –no como las dos que necesita el programa–. Así pues, estas rutinas están diseñadas para limpiar solamente el área colocada aparte de la ventana de texto.

El próximo artículo sobre los juegos de guerra trata sobre el mapa y el movimiento de las diferentes unidades dentro de él.

MICROCONSEJO: UN PLAN DETALLADO

Habiendo decidido los principales límites de tu juego de guerra, es muy importante insertar correctamente los detalles. Tú podrás decidir sobre todos los detalles que son relevantes para el combate en tu juego. Por ejemplo, podrías querer distinguir entre una guerra de misiles (si se trata de arco y flechas o de bombas de neutrones) y escaramuzas cuerpo a cuerpo. En muchos casos necesitarás como mínimo, dos tipos diferentes de misiles, dos tipos diferentes de tropas y alguna representación del recorrido. Como las armas de misiles serán menos efectivas a largas distancias, se hace importante conocer con qué rapidez (o cuán lejos) pueden moverse los dos tipos de tropas. Esto llevará a tener que considerar las armaduras. Las armaduras pesadas frenan a las tropas, pero estarán mejor protegidas. Las corazas no sólo afectarán al combate de misiles, sino también al combate cuerpo a cuerpo.

El combate también puede estar afectado por el terreno (el combate entre trincheras es más duro que el combate a campo abierto), por la protección (es más difícil acertar a un blanco detrás de un muro de un castillo que a uno detrás de un arbusto), por el número de tropas combatientes, etc.

Proyectando en este estadio, es muy importante tener en cuenta que ciertamente no siempre te será posible incluir todo lo que quieras en tu juego, aun cuando dispongas de una gran cantidad de memoria, de forma que tendrás que hacer balance entre lo principal de tu juego y la cantidad de memoria RAM que poseas.

JUGANDO A LA GUERRA (y II)

- MAPAS, HOMBRES, ORDENES Y BATALLAS
- DESARROLLO DE LA ACCION
- FACTORES QUE AFECTAN AL MOVIMIENTO DE LAS TROPAS

En el capítulo anterior se establecieron los símbolos de cada una de las unidades militares necesarias para el combate. Éstos se representan (y mueven) sobre el mapa para indicarnos el progreso y estrategia del juego.

El programa posee unas variables para representar el mapa. Como éste se visualiza durante todo el tiempo, dichas variables son innecesarias. Siempre habrá un área de memoria reteniendo la pantalla con toda la información.

Las variables tienen tantos elementos como posiciones de pantalla hay dentro del mapa. Una variable normal necesita unos cinco bytes por elemento, y una entera, cuatro bytes.

LAS TROPAS

Para identificar la situación de las unidades en el mapa (para que puedan ser movidas y detenidas si hay obstáculos en el camino), se necesitará otra variable: la variable de tropa. Dicha variable contiene la información que se precise para el combate, estado de ánimo, movimiento, etc.

FIJANDO LAS VARIABLES

Las siguientes rutinas dimensionan las variables correspondientes al mapa y a la tropa:

```
350 DIM M(16,27)
355 DIM T(16,9)
```

RELLENANDO EL MAPA

El siguiente paso consiste en determinar el terreno y las posiciones de partida de cada unidad y mostrarlas en pantalla.

El modo más sencillo para ello es utilizar el generador de números aleatorios del ordenador.

ESCOGIENDO EL TERRENO

La rutina de **elección de terreno** es esencialmente aleatoria, pero existe un grado de control sobre la elección. Esto asegura que el terreno sea trazado de forma realista con bosquecillos, colinas, ríos, etc.

```
17 CG=RND(-TIME): CG=(CG*
   10)
18 FOR FF=1 TO CG
20 DEF FN R(X)=INT(RND(1)*
   X)+1
21 NEXT
```

BAJO CONTROL

Veamos la rutina de elección de terreno en la que se va a desarrollar la batalla.

```
370 I$="nose"
470 REM ** CREANDO **
475 FOR XF=1 TO 15
476 LOCATE 1,XF:
    PRINT"FFFFFFFFFFFF
    FFFFFFFFFFFFFFF"
477 NEXT
480 FOR I=1 TO 16: GOSUB 800:
    M(I,1)=R: NEXT I
490 FOR I=1 TO 15
500 FOR J=2 TO 27
510 S=FN R(10)
520 IF S<8 THEN GOSUB 800
525 IF S>=8 THEN R=M(I,J-1)
530 M(I,J)=R
540 IF R=3 AND J<27 THEN
    M(I,J+1)=4
550 IF M(I,J)<>0 AND M(I,J)<>3
    THEN LOCATE J,I:
    PRINTCHR$((M(I,J)+64))
555 IF M(I,J)=3 AND J<>27 THEN
    LOCATE J,I: PRINTCHR$(67):
    LOCATE J+1,I:
    PRINTCHR$(68)
560 NEXT J
570 NEXT I
585 GOSUB 720
590 FOR I=1 TO 8
600 FOR J=1 TO 2
605 T(I+8,J)=2: NEXT J
610 FOR J=3 TO 4: READ T (I,J)
620 READ MR
630 FOR J=0 TO 8 STEP 8
640 T(I+J,5)=MR+FNR (2)
650 T(I+J,6)=(FN R(100)*10)+10
660 T(I+J,7)=T(I+J,6)
670 NEXT J
680 T(I,8)=15
690 T(I+8,8)=1
700 NEXT I
705 IF TT1=2 THEN GOSUB 2900
706 IF TT1=3 THEN GOSUB 2910
707 IF TT1=4 THEN GOSUB 2920
710 RETURN
```

DESARROLLO DE LAS TROPAS

Las posiciones de las tropas están contenidas en la variable de tropa como un par de coordenadas: horizontal y vertical.

Las posiciones de partida de las unidades oponentes se hallan en diferentes extremos del mapa: la posición inicial del jugador en el extremo sur, y la del ordenador, en el extremo norte.

```
860 REM ** DISPONER LAS
    TROPAS **
870 REM
880 FOR M=1 TO 2
890 S=1: R=1
900 FOR K=1 TO 8
910 REM
920 S=FNR(8*M)
930 IF T(S,9)<>0 THEN GOTO
    910
940 R=FN R(4)+R
950 R=R- INT(R/27)
960 T(S,9) = R
970 REM
980 LOCATE T(S,9),T(S,8)
985 PRINTMID$ (U$,S,1)
990 NEXT K
1000 NEXT M
1010 RETURN
```

SOBRE EL CAMPO DE BATALLA

Una vez que han sido determinadas las posiciones de partida de ambos ejércitos, las unidades están preparadas para ser mostradas en pantalla.

```
410 U$=CHR$(72)+CHR$(73)+
    CHR$(74)+CHR$(74)+
    CHR$(75)+CHR$(75)+
    CHR$(76)+CHR$(76)
411 U1$=CHR$(80)+CHR$(81)+
    CHR$(82)+CHR$(82)+
    CHR$(83)+CHR$(83)+
    CHR$(84)+CHR$(84)
412 U$=U$+U1$
413 VPOKE8201,143:VPOKE
    8196,16:VPOKE8218,240
414 VPOKE8202,79:VPOKE
    8217,240
```

EL RECINTO HOSTIL

La pantalla puede volverse más atractiva si se dibuja una frontera limitando el campo de batalla.

```
720 REM **DECORACION DEL
    BORDE **
730 FOR I=0 TO 16
740 LOCATE 0,I: PRINT CHR$(69)
741 LOCATE 28,16-I:PRINTCHR$(69)
750 NEXT I
760 FOR I=0 TO 28
770 LOCATE I,0: PRINT CHR$(69)
771 LOCATE 28-I,16: PRINT
    CHR$(69)
780 NEXT I
790 RETURN
```

MOVILIZANDO LAS FUERZAS

Ahora que el mapa está trazado, ambas partes desearían mover sus ejércitos. Estos se desplazan respondiendo a otras órdenes, pero hay diversos presupuestos que deben ser examinados antes de que una unidad pueda ser movida dentro del campo de batalla.

El programa debe saber cuál es el máximo movimiento (número de cuadrados) para cada unidad. La movilidad de éstas depende únicamente del peso del armamento, pero también podrían influir en ella la disciplina, la moral, el cansancio...

Con la siguiente rutina el programa podrá comprobar dichos factores antes de poner en movimiento alguna unidad.

```
1160 REM **MOVIMIENTO DE LA
     UNIDAD **
1170 OX=T(B,8): OY=T(B,9)
1175 Z$=" "
1180 IF M(T(B,8),T(B,9))<>0
     THEN Z$=CHR$(64+M
     (T(B,8),T(B,9)))
1190 D=5- T(B,4)
1200 IF B<3 OR B=9 OR B=10
     THEN D=D+2
1210 V=T(B,2)-1
1215 UP=0: AL=V-2
1220 IF (V/2-(INT(V/2)))=0 THEN
     UP=V-1: AL=0
1230 REM
1240 N1=T(B,9)+AL:
     NP=T(B,8)+UP
1250 IF NP<1 THEN NP=1
1260 IF NP>15 THEN NP=15
1270 IF N1<1 THEN N1=1
1280 IF N1>27 THEN N1=27
1290 IFM(NP,N1)>0THEND=D-1
1300 FOR K=1 TO 8
1310 IF (T(K,9)=N1 AND
     T(K,8)=NP ANDK<>B)
     THEN D=0
1315 IF (T(K+8,9)=N1 AND
     T(K+8,8)=NP AND
     K+8<>B) THEN D=0
1320 NEXT K
1330 IF D>0 THEN T(B,9)=N1:
     T(B,8)=NP: D=D-1
1340 IF D<>0 THEN GOTO 1230
1350 LOCATE OY,OX: PRINTZ$
1355 IF Z$=" " THEN LOCATE
     OY,OX;PRINT"F"
1360 LOCATE T(B,9),T(B,8):
     PRINTMID$(U$,I,1)
1370 RETURN
```

EL ARTE DE COMANDAR

Hay un número de factores conectados con las tropas que afectarán a la manera en que éstas actúen en el campo de batalla. Veamos algunos de ellos:

–Orden actual de la unidad (lo último que se les ordenó que hicieran);
–Dirección del movimiento actual;
–Armamento y armaduras;
–Fuerza: inicial y actual;
–Moral o actitud;
–Posición y terreno.

Estos factores corresponden a los elementos de la variable de tropa que establecimos anteriormente.

VISITANDO EL CUARTEL GENERAL

Esta rutina inicializa los elementos no completados de la variable:

```
190 REM ** INICIALIZACION **
200 VC=0: DE=0
310 REM ** definición de matrices
    **
320 COLOR ,15,6
330 COLOR 1
340 CLS
360 DIM T$(8,12): DIM O$(5,12):
    DIM W$(5,9)
361 DIM M$(5,12): DIM A$(4,12):
    DIM R$(4,12): DIM C(8)
385 READ O1$: READ W1$: READ
    M1$
386 O$(I,12)=O1$: W$(I,9)=W1$:
    M$(I,12)=M1$
387 NEXT I
389 RESTORE2790
390 FOR I=1 TO 8
395 READ T1$: T$(I,12)=T1$
396 NEXTI
399 RESTORE2800
400 FOR I=1 TO 4
405 READ A1$: READ R1$
406 A$(I,12)=A1$: R$(I,12)=R1$
407 NEXT I
415 X$="NnSs"
420 RETURN
2760 DATA "fuego","ningun",
     "cobarde","alto","arco",
     "indispuesta"
2770 DATA "movimiento",
     "espada","dispuesta",
     "estado","hacha","brava"
2780 DATA "derrota","lanza",
     "valiente"
2790 DATA "caballeros",
     "sargentos","lanceros",
     "lanceros","arqueros",
     "arqueros","aldeanos",
     "aldeanos""
2800 DATA "ningun","llanuras",
     "chaquet"n","pueblo",
     "cotaFmalla","bosque",
     "mesetas","colinas"
2810 DATA 5,4,3,5,3,3,4,3,2,3,3,1,
     2,2,1,2,3,2,3,2,0,3,1,0
```

DANDO ORDENES

La totalidad del juego depende de las órdenes dadas por el jugador al ejército; sin éstas no habría combate y, por consiguiente, ni vencedores ni vencidos.

Hay cuatro órdenes que podemos dar a las tropas: fuego, hacer alto, movimiento y status. La orden de abrir fuego sólo se aplica a los arqueros, y la cumplirán aun cuando no exista un blanco en los alrededores. Si seleccionamos abrir fuego, y la unidad no es de arqueros, entonces se nos solicitará otra orden; la orden de hacer alto es explícita por sí misma; la orden de movimiento solicita la entrada de una determinada dirección; en lo referente al status, éste responde absolutamente a la descripción del estado actual de la unidad.

SELECCIONANDO UNA UNIDAD

Cuando es el turno del jugador, la primera unidad se ilumina, seguida en turno por las otras siete.

```
1380 REM ** Selecciona la unidad
     para órdenes **
1390 GOSUB 2540
1400 REM
1401 LOCATE 7,21: PRINT"
```



```
        pulsaFunaFtecla"
1403 LOCATE T(I,9),T(I,8):
        PRINT"F"
1405 FOR TL=1 TO 50:NEXT TL
1410 LOCATE T(I,9),T(I,8):
        PRINTMID$(U$,I,1)
1411 FOR TJ=1 TO 50:NEXT TJ
1415 LET G1$=INKEY$: IF G1$=""
        THEN  GOTO 1403
1417 GOSUB 2540
1418 FOR LP=18 TO 20: LOCATE
        0,LP
1419 PRINT"FFFFFFFFFFFFFFFFF
        FFFFFFFFFFF": NEXT LP
1420 LOCATE 0,18: PRINT"unidad
        FnumeroF"+STR$(I)+"F"+T$
        (I,12)+"FFFF"
1425 LOCATE 14,18: PRINT"F"
1430 LOCATE 0,19:
        PRINT"lasFordenesFson";
        "F";O$(T(I,1),12)
1440 IF T(I,1)=3 YHEN LOCATE
        28,19: PRINTMID$(I$,T(I,2),1)
1450 REM ** Variables para el
        ciclo **
1460 LOCATE 0,20: PRINT
        "cambiaFlasFordenesFF(s/n)
        ?"
1465 Y=0: Y$=INKEY$
1466 IF Y$="" THEN GOTO 1465
1470 FOR K=1 TO 4
1475 IF MID$(X$,K,1)=Y$ THEN
        Y=K
1480 NEXT K
1485 IF X$(K)=Y$ THEN Y=K
1490 IF Y=0 THEN GOTO 1450
1500 RETURN
```

EMITIENDO ORDENES

Esta rutina muestra las órdenes opcionales.

```
1900 REM ** Selecciona   accion
        **
1910 GOSUB 2540
1920 LOCATE 1,18:
        PRINT"opcionesFson:"
1930 FOR J=1 TO 4
1935 LOCATE 15,17+J:
        PRINT"FFFFFFFFFFFF"
1940 LOCATE 15,17+J:
        PRINTLEFT$(O$(J,12),1);
        "-";O$(J,12)
1950 NEXT J
1960 REM ** Simulacro de bucle
        **
1962 A=0
1965 F$="FfAaMmEe"
1970 G$=INKEY$: IFG$=""THEN
        GOTO 1970
1975 FOR K=1 TO 8
1980 IF MID$(F$,K,1)=G$ THEN
        A=INT((K+1)/2)
1985 NEXTK
1990 IF A<=0 THEN GOTO 1960
2000 IF I<>6 AND I<>5 AND
        A=1  THEN GOSUB 2540:
        LOCATE 10,18:
        PRINT"ningunFarco":
        GOSUB 2410: GOTO 1910
2010 IF A=4 THEN GOSUB 2440:
        RETURN
2020 T(I,1)=A
2030 IF A=3 THEN GOSUB 2050
2040 RETURN
```

UNA NUEVA DIRECCION

Si el jugador emite una orden de movimiento, deberá asignársele también a ésta una dirección. Esta rutina maneja las opciones de movimiento.

```
2050 REM **Decide la dirección
        del movimiento **
2055 GOSUB 2540
2060 LOCATE 0,19: PRINT"
        queFdirecci"n:"
2061 LOCATE 20,18: PRINT"n"
2062 LOCATE 20,17:
        PRINTCHR$(206)
2063 LOCATE 18,19:
        PRINTCHR$(208)+"o "
2064 LOCATE 21,19:  PRINT"e"+
        CHR$(207)
2065 LOCATE 20,20: PRINT"s"
2066 LOCATE 20,21:
        PRINTCHR$(205)
2068 G=0
2070 REM ** Simulacro para
        bucle **
2080 G$=INKEY$: IF G$=""THEN
        GOTO 2080
2090 IF ASC(G$)<96 THEN
        G$=CHR$(ASC(G$)+32)
2095 FOR K=1 TO 4
2100 IF MIF$(I$,K,1)=G$ THEN
        G=K
2105 NEXT K
2110 IF G=0 THEN GOTO
        2070
2120 T(I,2)=G
2130 RETURN
```

STATUS: ESTADO

Podemos saber el estado de cualquier unidad, mientras planeamos el gran ataque, seleccionando la opción status.

Serán mostrados todos los elementos de la variable de tropa (o sus equivalentes en palabras).

```
2440 REM ** ESTADO DE LA
        UNIDAD **
2450 GOSUB 2540
2460 LOCATE 0,17:
        PRINT"unid.:"+STR$(I)
        +"FFFFtipo.:"+T$(I,12)+
        "FFFFFFF"
2465 LOCATE 6,17: PRINT"F"
2470 LOCATE 0,18: PRINT"arma.:
        "+W$(T(I,3),9)+
        "FFFFFFFFFF"
2480 LOCATE 12,18: PRINT"prtc.:
        "+A$(T(I,4),12)+
        "FFFFFFFFFF"
2490 LOCATE 0,19: PRINT"frza.:
        "+STR$(T(I,7))+
        "FFFFFFFFF"
2495 LOCATE 6,19: PRINT"F"
2500 LOCATE 12,19: PRINT"acti.:
        "+M$(T(I,5),12)+
        "FFFFFFFFF"
2510 LOCATE 0,20: PRINT"locz.:
        "+R$((M(T(1,8),T(I,9))+1),12)
        +"FFFFFFFFFFFFFFF"
2520 GOSUB 2410
2530 GOTO 1900
```

EL EFECTO DE LAS ORDENES

La rutina siguiente comunica al jugador si una determinada unidad está cumpliendo la orden encomendada.

```
1020 REM **EFECTO DE LAS
        ORDENES **
1030 FOR I=1 TO 16
1032 IF T(I,1)>3 THEN GOTO 1140
1035 GOSUB 2540: LOCATE 2,19
1037 PRINT"unidadF"+STR$(I)
        +"FdecidesFactuar"
1038 LOCATE 9,19: PRINT"F"
1040 CL=1: IF I>8 THEN CL=2
1050 IF T(I,1)=3 THEN B=I:
        GOSUB 1160
1055 IF T(I,1)=2 THEN GOTO 1140
1060 IF T(I,1)=1 THEN SH=I:
        GOSUB 1710: GOTO 1140
```

```
1070 FOR F=-1 TO 1
1080 FOR G=-1 TO 1
1090 FOR E= 1 TO 16
1100 IF (T(I,8)+F=T(E,8))   AND
     (T(I,9)+G=T(E,9)) AND
     T(E,1)<>5 THEN US=I:
     TH=E: GOSUB 1510
1110 NEXT E
1120 NEXT G
1130 NEXT F
1140 NEXT I
1150 RETURN
```

EL ORDENADOR COMO OPONENTE

El ordenador da las órdenes a sus unidades de un modo más o menos arbitrario.

```
2140 REM** EL ENEMIGO
     SELECCIONA LA ACCION **
2150 T(E,2)=3
2160 T(E,1)=FN R(3)
2170 IF T(E,1)=1 AND T(E,3)<>2
     AND  T(E,1)<>5 THEN
     GOTO 2160
2180 IF T(E,1)=3 AND T(E,1)<>5
     THEN IF FN R(2)=1 THEN
     T(E,2)=FN R(4)
2190 RETURN
```

PROYECTILES

Existen dos clases diferentes de combate: proyectiles (flechas) y cuerpo a cuerpo. Esta primera rutina trata el combate con proyectiles.

```
1710 REM ** RUTINA DE
     PROYECTILES **
1720 GOSUB 2540
1730 LOCATE 4,18:
     PRINT"laFunidadF"+
     STR$(SH)+F"disparo"
1735 LOCATE 14,18: PRINT"F"
1740 FX=5:
     FY=5: GP = -1
1745 ST=9
1750 IF SH>8
     THEN ST=1
1770 FOR M=ST
     TO (ST+7)
1780 TM=ABS
     (T(M,8)-T(SH,8)):
     TY=ABS
     (T(M,9)-T(SH,9))
1785 IF TM<FX AND T(M,1)
     <5 AND TY<FY THEN
     FX=TM:
     FY=TY: GP=M
1790 NEXT M
1800 IF GP=-1
     THEN LOCATE 7,19: PRINT
     "nadaFalFalcance":
     GOSUB 2410: RETURN
1810 C=8-T
     (GP,4)-ABS(FX-FY)
1820 IF GP<3
     OR GP=9 OR GP=10  THEN
     C=C+1
1830 IF M(T(GP,8),
     T(GP,9))=2 THEN
     C=C-2
1840 IF T(GP,1)
     <>2 THEN
     C=C+1
1850 C=
     (C+(INT(T(SH,7)/40))
     +FN  R(3))*10
1860 T(GP,7)=
     T(GP,7)-C
1868 LOCATE 0,
     20: PRINT"FFFFFF
     FFFFFFFFFFFFFFFFFFFFFFF"
1869 LOCATE 0,19:
     PRINT"FFFFFFFFFFFF
     FFFFFFFFFFFFFFFFF"
1870 LOCATE 0,19:
     PRINT"seFhanFproducido:"
     +STR$(C)+
     "FbajasFFFFFFFF"
1871 LOCATE 17,19:
     PRINT"F"
1872 LOCATE   0,20:
     PRINT"enFlaFunidad:"+
     STR$(GP)
1873 LOCATE 13,20:
     PRINT"F"
1875 GOSUB 2410
1880 UN=GP:
     UM=GP: GOSUB 2200
1890 RETURN
```

EL CHOQUE

El combate cuerpo a cuerpo se calcula de una forma similar.

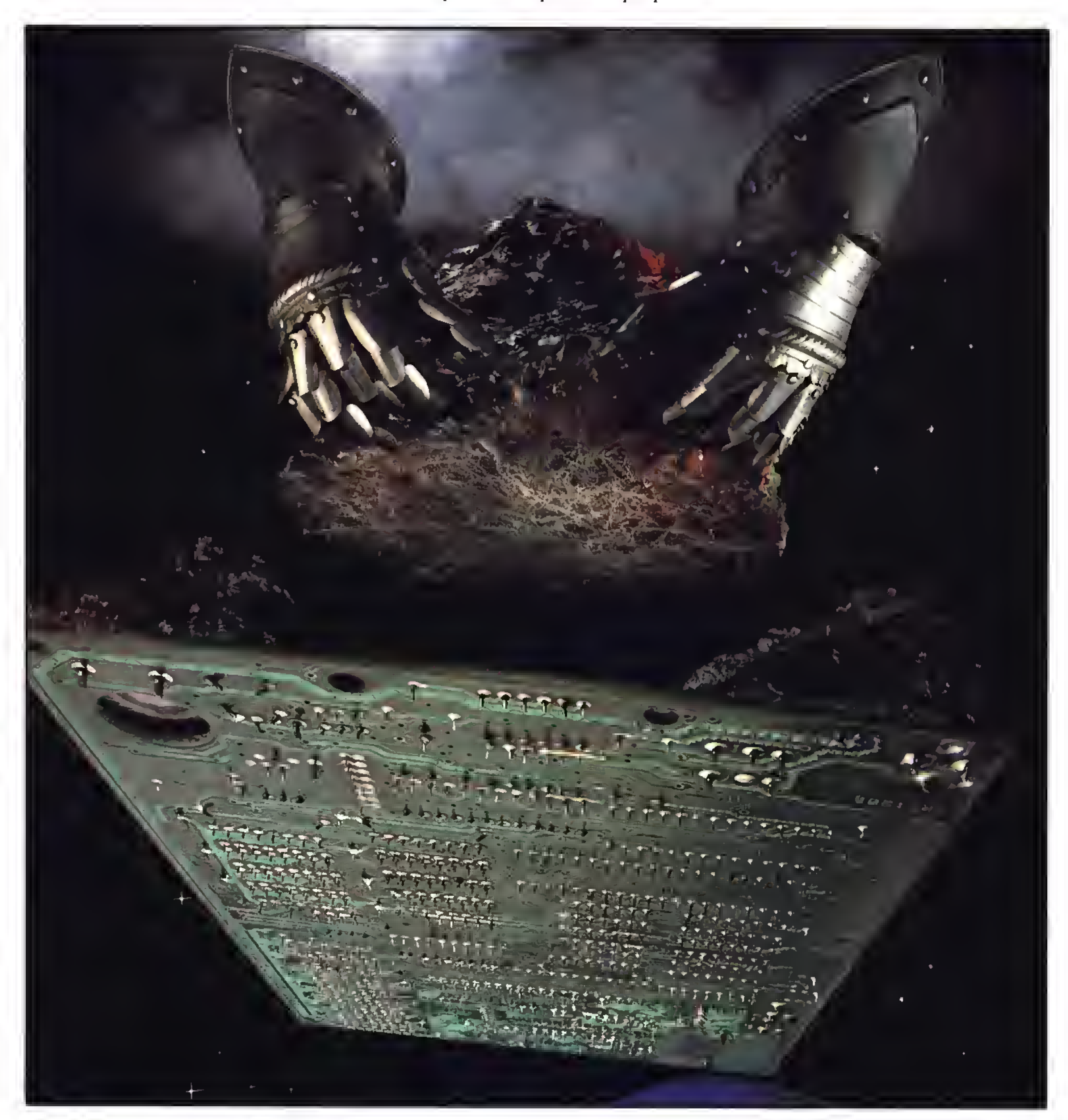
```
1510 REM ** COMBATE **
1520 IF (US<9 AND TH<9) OR
     (US>8 AND TH>8) THEN
     RETURN
1530 IF T(US,1)=5 OR T (TH,1)=5
     THEN RETURN
1540 GOSUB 2540
1550 LOCATE 8,18
1551 PRINT"——combate——"
1560 AT=INT((T(US,7)-T
     (TH,7))/50)
1570 AT=AT+T(US,3)-T(TH,4)+
     T(US,5)+FN R(5)
1580 IF ABS(T(US,2)-T(TH,2))<>
     2 THEN AT=AT+2
1590 IF US<3 OR US = 9 OR
     US=10 THEN AT=AT+1
1600 DR=INT((T(TH,7)-T(US,7))
     /60)
1610 DR=DR+T(TH,3)-T(US,4)-
     T(TH,5)+M(T(TH,8),T(TH,9))
     +FN R(3)+2
1615 WN=TH: LO=US
1620 IF AT>DR THEN WN=US:
     LO=TH
1630 WC=INT(T(WN,7)/10)
1631 IF WC<1 THEN WC=1
1640 T(WN,7)=T(WN,7)-WC
1650 LC=INT(T(LO,7)/5)
1651 IF LC<1 THEN LC=1
1660 T(LO,7)=T(LO,7)-LC
1665 FOR FK=19 TO 20: LOCATE
     0, FK
1667 PRINT"FFFFFFFFFFFFFFFFFF
     FFFFFFFFF":NEXT FK
1670 LOCATE 0,19:
     PRINT"unidad:"+STR$(WN):
     LOCATE 15,19: PRINT
     "perdidas:"+STR$(WC)
1672 LOCATE 7,19: PRINT"F":
     LOCATE 24,19: PRINT"F"
1675 LOCATE 0,20: PRINT"
     unidad:"+STR$(LO):
     LOCATE 15,20: PRINT
     "perdidas:"+STR$(LC)
1677 LOCATE 7,20: PRINT"F":
     LOCATE 24,20: PRINT"F"
1680 GOSUB 2410
1690 UN=LO: UM=WN: GOSUB
     2200
1700 RETURN
```

La principal diferencia entre el combate cuerpo a cuerpo y el combate de proyectiles reside en que ambas partes tienen algo que decir en la afrenta. Esto significa que el combate cuerpo a cuerpo es un asunto más desordenado, que requiere dos juegos de bajas para ser calculado. A continuación, la diferencia entre las armas de los atacantes y el armamento de los

defensores es agregada, junto con el valor de la moral de los atacantes y un número aleatorio superior a cinco.

Los atacantes también ganan una bonificación si el enemigo no se enfrenta directamente a ellos. Este será el caso, tanto si la unidad no está moviéndose directamente hacia los atacantes, como –cuando la unidad ha hecho alto– si no estaba moviéndose en aquella dirección en la que se movía la última vez. Ello se debe a que el elemento de dirección de la variable de tropa nunca es eliminado, y siempre debe poseer una dirección. Históricamente, parece ser que el ataque por la retaguardia o por los flancos ha sido siempre uno de los factores más significativos del combate. Finalmente, los atacantes obtienen una bonificación de un punto si pertenecen a la caballería.

Los defensores son tratados similarmente. El factor arbitrario trata de de-



terminar el hecho de que es más fácil defender que atacar, pero algunas veces el ansia de sangre de los atacantes puede intervenir para predominar sobre la ventaja innata del hecho de defender.

TEST MORAL DE LA UNIDAD

El factor psicológico es muy importante en la guerra. Es bastante improbable que un ejército gane una batalla, aun equipado con los más poderosos carros de combate del mundo, si las tropas odian a sus generales, si creen que el enemigo tiene una causa justa, o si les repugna la idea de combatir, sea como sea. Existen miles de factores involucrados en la psicología individual de los soldados, y ningún juego ha llegado nunca a representarlos todos en su descarnada realidad. En el nivel más ridículo, la actitud de un guerrero en una pugna corriente puede depender de si ha dormido bien o no la noche anterior, o de la calidad de su última comida.

La moral no sólo puede ser complicada, sino que puede influir justamente sobre cada uno de los aspectos de la batalla, dando un empuje o bloqueando a los bandos. La moral sólo pasa el test cuando una unidad pierde un combate o es arrasada.

```
2200 REM ** TEST MORAL DE LA
     UNIDAD **
2205 GOSUB 2540
2206 LOCATE 1,18:
     PRINT"unidad:
     F"+STR$(UN),"Fhombres:F"
     +STR$(T(UN,7))
2207 LOCATE 1,19: PRINT"unidad
     :F"+STR$(UM),"Fhombres:
     F"+STR$(T(UM,7))
2210 GOSUB 2410
2215 IF T(UN,7)>1 AND
     T(UM,7)>1 THEN GOSUB
     2540: RETURN
2220 GOSUB 2540
2230 LOCATE 1,18: PRINT"perdidas
     FdemasiadoFgrandes"
2240 LOCATE 1,19: PRINT"
     unidad:"+STR$(UN)+
     "Fdesintegrada"
2245 LOCATE 8,19: PRINT"F"
2250 GOSUB 2410
2260 IF T(UN,7)<=1 THEN T
     (UN,1)=5:GOTO 2270
2265 IF T(UM,7)<=1 THEN
     T(UM,1)=5:GOTO 2275
2270 LOCATE T(UN,9),T(UN,8):
     PRINT"F"
2272 RETURN
2275 LOCATE T(UM,9),T(UM,8):
     PRINT"F"
2280 RETURN
2290 REM ** Test para victoria **
2300 GD=0: BD=0
2310 FOR M=1 TO 8
2320 IF T(M,1)<>5 THEN
     GD=GD=1
2330 IF T(M+8,1)<>5 THEN
     BD=BD+1
2340 NEXT M
2350 IF BD<1 AND GD>0 THEN
     VC=1
2360 IF GD<1 AND BD>0 THEN
     DE=1
2370 RETURN
```

EL ARMISTICIO

Hay que añadir unos pocos toques finales al programa: en primer lugar una condición de victoria.

```
2380 REM **Fin del mensaje **
2390 IF VC=1 THEN GOSUB 2950:
     GOSUB 2930
2395 IF DE=1 THEN GOSUB 2950:
     GOSUB 2940
2400 RETURN
2410 REM ** RETRASO **
2420 LOCATE 7,21: PRINT
     "pulsaFunaFtecla"
2425 G$=INKEY$: IF G$="" THEN
     GOTO 2425
2430 RETURN
2930 CLS: FOR FO=1 TO 16
2931 PRINT"——victoria——"
2935 NEXT: RETURN
2940 CLS: FOR FO=1 TO 16
2941 PRINT"unFenfrentamientoF
     frustrado"
2945 NEXT: RETURN
2950 FOR F=1 TO 23
2951 PRINT"
2952 NEXT: RETURN
```

Ahora ya tienes todas las rutinas que completan el *wargaming*. Todo lo que necesitas para gobernar a tus tropas es el bucle principal.

```
10 CLEAR 600
12 COLOR 15,1,2
15 SCREEN1:GOSUB 3000
30 GOSUB 190
35 GOSUB 3200
40 GOSUB 470
50 GOSUB 860
60 REM ** SIMULACRO PARA
   REPETICION DE BUCLE **
70 FOR I=1 TO 8
80 IF T(I,1)<4 AND T(I,1)<>5
   THEN GOSUB 1380
85 IF Y>2 AND T(I,1)<>5 THEN
   GOSUB 1900
90 IF T(I,1)<5 AND T(I,1)<>5
   THEN LOCATE T(I,9),T(I,8):
   PRINT MID$(U$,I,1)
100 NEXT I
110 FOR E=9 TO 16
120 IF T(E,1)<4 THEN GOSUB
    2140
130 NEXT E
140 GOSUB 1020
150 GOSUB 2290
160 IF VC<>1 AND DE<>1 THEN
    GOTO 60
170 GOSUB 2380
180 GOSUB 2410: RUN
205 RESTORE 2570
210 FOR I=520 TO 567
215 READ B
220 VPOKE I,B
225 NEXT I
230 FOR I=576 TO 615
240 READ BA
250 VPOKE I,BA
260 VPOKE I+64,BA
290 NEXT I
800 REM
810 R=FNR (50)
820 IF R>5 THEN R=0
830 IF R>4 THEN R=3:RETURN
840 IF R>1 THEN R=2
850 RETURN
2570 DATA 16,16,60,126,255,189,
     231,231
2590 DATA 16,56,84,16,56,84,146,
     16
2610 DATA 8,20,34,65,6,8,16,224
2630 DATA 0,48,72,132,2,0,0,0
2635 DATA 255,231,231,129,129,
     231,102,60
2636 DATA 0,0,0,0,0,0,0,0
2650 DATA 128,240,255,252,143,
     128,128,128
2670 DATA 64,240,72,68,68,68,78,
     68
```

```
2690 DATA 255,231,231,129,129,
     231,102,60
2710 DATA 249,70,38,25,9,5,3,1
2730 DATA 1,2,4,8,16,160,64,160
2900 FOR FZ=1 TO 8
2905 T(FZ+8,7)=INT(T(FZ+8,7)
     *1.5)
2907 NEXT: RETURN
2910 FOR FZ=1 TO 8
2915 T(FZ+8,7)=T(FZ+8,7)*2
2917 NEXT: RETURN
2920 FOR FZ=1 TO 8
2925 T(FZ+8,7)=T(FZ+8,7)*3
2929 NEXT: RETURN
3000 GOSUB 3030
3001 LOCATE 5,1: PRINT"WAR
     GAMING"
3002 LOCATE 8,5: PRINT"1–
     FACIL"
3003 LOCATE 8,9: PRINT"2–
     NORMAL"
3004 LOCATE 8,13: PRINT"3–
     DIFICIL"
3005 LOCATE 8,17: PRINT"4–
     IMPOSIBLE"
3006 FOR FA=3 TO 19
3007 LOCATE 6,FA:
     PRINTCHR$(199)
3008 LOCATE 21,FA:
     PRINTCHR$(199)
3009 NEXT
3010 FOR FA=7 TO 20
3011 LOCATE FA,3:
     PRINTCHR$(199)
3012 LOCATE FA,19:
     PRINTCHR$(199)
3013 NEXT
3014 T2T=2
3015 ID T2T=1 THEN VPOKE
     8216,111: T2T=2: GOTO
     3020
3016 IF T2T=2 THEN VPOKE
     8216,1: T2T=1
3020 LET G2$=INKEY$: IF G2$="
     " THEN GOTO 3015
3021 IF ASC (G2$) < 49 OR ASC
     (G2$)>52 THEN GOTO 3020
3022 LET TT1=VAL(G2$)
3023 CLS: RETURN
3030 CLS: RESTORE 3604: FOR
     N=1 TO 29
3031 READ Q: Q$=Q$+CHR$(Q):
     NEXT N
3035 PRINT: FOR N+1 TO
     7:PRINTQ$: PRINT: NEXT N
3036 FOR N=1 TO 400: NEXT N
3040 CLS: RETURN
3200 FOR FD=1 TO 16
3210 LET T(FD,1)=2
3220 NEXT FD
3230 RETURN
3604 DATA 42,98,121,32,32,68
3605 DATA 65,78,73,69,76,32,67,
     65
3606 DATA 76,86,69,84,32,76,69,
     80
3607 DATA 69,75,72,73,78,69,42
     8192  ,192
```

INSTRUCCIONES DEL JUEGO

Inmediatamente que ejecutamos (RUN) el programa, el ordenador se pone a dibujar el mapa. Los símbolos del terreno aparecen en primer lugar, seguidos de las unidades oponentes y de la frontera.

Ahora es el momento de empezar a construir la estrategia. Una serie de símbolos aparecerán en la «ventana» del texto. Comenzando desde la unidad uno, el número de unidad y la descripción –por ejemplo, caballeros– junto con órdenes actuales, **Hacer alto,** tal vez. Al jugador se le pregunta **Cambio (S/N)?.**

Si la respuesta es S, se muestra un menú de órdenes opcionales: **Abrir fuego, Hacer alto** o **Moverse.** La opción de **Abrir fuego** sólo queda abierta para los arqueros; así pues, cualquier intento de hacer que otro tipo de unidad abra fuego, hará aparecer el mensaje «Ningún arco», y el ordenador esperará otra elección. Si se selecciona la opción de movimiento, el apunte **Qué camino?**, (N,S,E,W) hará su aparición, listo para la elección del jugador.

